VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2011 Lukáš Kuna

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Sada utilit pro IPTV vysílání z DVB zdrojů

Utilits for IPTV Broadcasting from DVB sources

2011 Lukáš Kuna

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 3. května 2011

Poděkování

Za odbornou pomoc a podporu při vypracování projektu bych chtěl na tomto místě poděkovat vedoucímu práce Ing. Davidovi Seidlovi.

Abstrakt

Televizní vysílání po internetovém protokolu (IPTV) představuje moderní způsob distribuce audiovizuálních dat. V rámci této práce jsem se pokusil analyzovat a řešit některé potřeby a problémy s tímto spojené.

Navrhnul a realizoval jsem funkční dekodér teletextových dat a informací týkajících se elektronického programového průvodce (EPG). Vytvořené aplikace představují nákladově a funkčně efektivní řešení daného zadání a umožňují rychlou integraci do systémů provozovatele převzatého televizního vysílání.

Pro případ kompletního výpadku některého ze zdrojů digitálního televizního signálu nebo jeho drobného porušení jsem vypracoval možnosti, jak tyto nežádoucí jevy potlačit a vytvořil jsem aplikace, které pokryjí podstatnou část potřeb v této oblasti u většiny IPTV provozovatelů.

Klíčová slova

DVB, IPTV, MPEG2-TS, EPG, teletext, ztrátovost, poškození, výpadek, čítač souvislosti, transport error indicator

Abstract

The internet protocol television (IPTV) represents a modern distribution method for audiovisual data. In this work, I have analyzed and solved some of the needs and troubles coupled with IPTV.

I have designed and prepared a functional decoder of the teletext data and other information related to electronic program guide (EPG). The established applications represent a cost- and function- effective solution of the task and enable a fast integration into the system of a telecast operator.

In case of a complete failure or little disruption of the digital signal supply, I have elaborated several options how to eliminate these unwanted features; I developed several applications, which cover a significant portion of the needs in the field of IPTV operators.

Keywords

DVB, IPTV, MPEG2-TS, EPG, teletext, loss, disruption, failure, continuity counter, transport error indicator

Seznam použitých zkratek

AV – Audio Video
BCD – Binary Coded Decimal
CRC – Cyclic Redundancy Check
ČS – čítač souvislosti
DTS – Decode Time Stamp
DVB - Digital Video Broadcasting
EIT - Event Information Table
EPG - Electronic Program Guide
ES - Elementary Stream
FEC - Forward Error Correction
FTTx - Fiber To The x
HTTP - HyperText Transfer Protocol
IP – Internet Protocol
IPTV - Internet Protocol TeleVision
JIT - Just In Time
MPEG-2 - Moving Picture Experts Group 2
MPEG2-TS - Moving Picture Experts Group 2 Transport Stream
OPCR - Original Program Clock Reference
PAT - Program Association Table
PCR - Program Clock Reference
PES - Packetized Elementary Stream
PID - Packet IDentifier
PMT - Program Map Table
PNR – Program NumbeR
PON - Passive Optical Network
PTS - Presentation Time Stamp
TCP - Transmission Control Protocol
TDS – TCP datový server
TEI - Transport Error Indicator
TV – TeleVision
UCS - Universal Character Set
UDP - User Datagram Protocol
UTF-8 - UCS Transformation Format – 8 bit
VBI - Vertical Blanking Interval
XML - eXtensible Markup Language

Obsah

1. Úvod

Zábava – činnost, která není nikomu cizí, ať v jakékoliv formě, přináší potěšení každému z nás. Zábavní průmysl v průběhu času doznal změn a rozrostl do velkých rozměrů, avšak od rozšíření televizních přijímačů v minulém století mohu bez obav označit televizní zábavu za oblast, které lidé věnují nezanedbatelnou část svého života.

Postupem doby muselo dojít k proměnám v technologiích týkajících se přenosu televizního signálu, významný přelom přinesla dnes tolik aktuální digitalizace vysílání. Spolu s ní však vyvstaly specifické problémy, objevily se nové přístupy a inovativní přístupy a techničtí vývojáři byli postaveni před mnohé výzvy, s kterými si museli a stále ještě musí poradit.

Tato práce si klade za cíl řešit dvě zájmové oblasti týkající se digitálního televizního vysílání:

- zpracování teletextových dat a dat elektronického programového průvodce, přesněji jeho dekódování a interpretace do člověkem srozumitelné podoby,
- odstranění poruch a výpadků přenosu digitálních dat televizního vysílání.

Zvláštní důraz by měl být kladen nejen na funkčnost celého řešení, ale především jeho vysokou stabilitu, protože nelze dopustit sebemenší výpadky ve vysílání, ale také na nízké technické a výkonnostní nároky na běh a co nejmenší závislosti na jiných řešeních, které by mohly ztěžovat vývoj a údržbu řešení nebo by mohly znamenat problémy při nasazení nebo jiné licenční náklady a překážky. Za samozřejmé lze považovat požadavky na dobrou rozšiřitelnost, modulárnost, údržbu, přehlednost a nenáročnou správu.

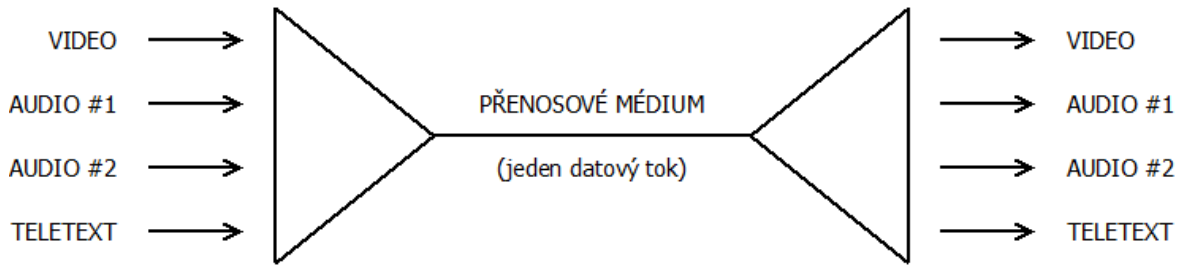
Zpracovávaná problematika nachází největší uplatnění v moderní metodě přenosu televizního vysílání IPTV – televize po IP protokolu, avšak použité technologie lze nalézt v ostatních populárních metodách za pomocí satelitního nebo pozemního příjmu založených na standardech skupiny DVB. Mezi cílovou skupinu uživatelů řešení tedy patří operátoři televizních kabelových rozvodů IPTV, kterým pomůže v efektivním vynaložení finančních prostředků, vylepšení kvality služby a rozšíření doplňkových funkcí, užitek však z části naleznou i zvídaví koncoví uživatelé příjmu digitální TV.

2. Architektura DVB vysílání

Živé vysílání IPTV se v dnešní době realizuje za pomocí technologií, které jsou přímo založeny na principech DVB, navíc zdrojem pro toto vysílání bývá velmi často pozemní a především satelitní vysílání DVB, proto má smysl si některé z těchto principů více či méně podrobně popsat, podle toho, jak to tato práce vyžaduje.

2.1 Vícesložkový proud dat

Každý televizní program obvykle obsahuje více elementární složek, typicky např. video, audio a teletext. Analogové vysílání umožňuje oddělený a současný přenos video a audio složky, avšak u digitálních přenosových metod se vysílají jednotlivé bity jeden za druhým, bez možnosti přímého paralelního příjmu (schematicky popisuje obrázek 1).

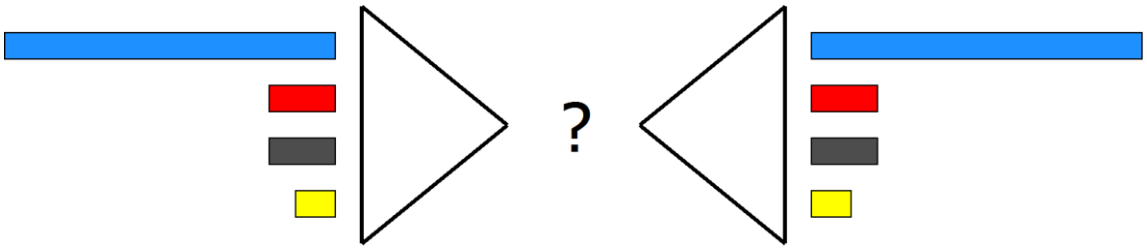


Obrázek 1: Typická ukázka přenosu jednoho digitálního televizního kanálu – video složka, dvě audio složky (různé jazykové mutace), teletext

2.1.1 Multiplexování

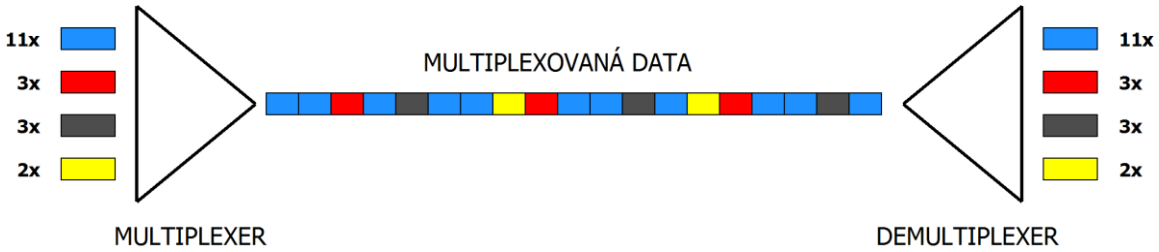
Princip, který umožní řešit problém současného vysílání více složek po jednom sdíleném médiu, se nazývá multiplexování. Zařízení nebo jiný technický prostředek, který provádí skládání více složek, se označuje jako multiplexor a ten, který se postará o rozložení zpět na jednotlivé složky, demultiplexor. Protože lze v jeden okamžik přenášet pouze jeden nebo několik málo bitů (podle použité modulace), jedná se o tzv. časový multiplex.

A jak tento princip funguje? Obecně se dá říci, že proudy jednotlivých složek multiplexor rozdělí na malé části (fragmenty) o velikosti několika bajtů (podle potřeb), jednotlivé části se označí tak, aby bylo zřejmé, ke které složce patří, a odesílají se střídavě po sdíleném médiu. Demultiplexor v přijímači pak tyto malé části postupně čte a podle toho, k jaké složce část patří, skládá souvislé toky jednotlivých složek. V případě digitálního vysílání byla zvolena pevná velikost těchto částí, na které se proud složek dělí.



Obrázek 2: Různé složky (každý má svou barvu) mají rozdílné požadavky na přenosovou kapacitu (znázorněny délkou proužku)

Vzhledem k tomu, že jednotlivé složky mívají velmi často rozdílné požadavky na přenosovou kapacitu (obrázek 2) – např. video obsahuje o mnoho více informací než audio, střídají se tyto fragmenty přibližně v poměru potřebné přenosové kapacity (obrázek 3).



Obrázek 3: Proud jednotlivých složek se rozdělí na fragmenty pevné délky a jejich vysílání se střídá v poměru požadavků na přenosovou kapacitu

Podrobnější informace o multiplexování lze nalézt např. v [5, strana 78 – 3.2].

2.1.2 Statistický multiplex

Jedním ze silných argumentů pro přechod na digitální televizní vysílání byla skutečnost, že dojde k efektivnějšímu využití frekvencí, které se pro přenos používají, a umožní se tak přístup většího množství stanic na televizní trh. Tento argument se stává platným z toho důvodu, že na frekvenci stejné šířky jako u analogového vysílání, lze za pomocí pokročilých modulačních technik přenášet tak velký datový tok, že vystačí hned pro několik televizních stanic. Skupině takto vysílaných kanálů se říká digitální multiplex.

V závislosti na parametrech použité modulace multiplexu lze efektivně využít různou celkovou šířku přenosového pásma, avšak tyto parametry se v čase nemění, takže dostupná šířka pásma zůstává také stále stejná. Provozovatelé multiplexů se snaží co nejefektivněji využít dostupné pásmo, takže často používají kompresi obrazu s proměnnou datovou rychlostí, neboli podle toho, jak je video scéna náročná, zvolí menší či větší část dostupné šířky pásma, takže stanice se složitějšími scénami dostanou přiděleno dočasně větší pásmo na úkor těch stanic, kde se např. vysílá statický obraz nebo scéna méně náročná. Tento princip, kdy se mění přidělená kapacita jednotlivým elementárním složkám na základě jejich potřeb, se nazývá statistický časový multiplex.

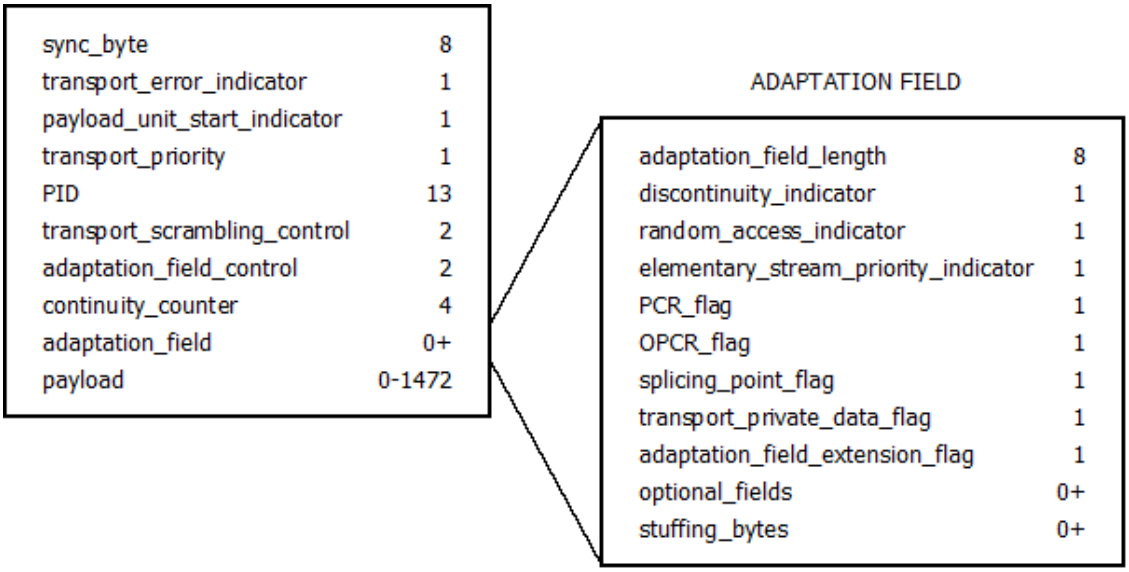
2.2 Formát MPEG2-TS

Přesný formát a pravidla, jak provádět multiplexování dat v DVB, popisuje norma ISO 13818-1 [1]. Vyhledem k tomu, že je norma součástí skupiny standardů MPEG-2, označuje se tento formát zkráceně jako MPEG2-TS, což vychází ze sousloví MPEG2 transport stream.

2.2.1 Paket

MPEG2-TS specifikuje délku fragmentu, tedy částí, na které se jednotlivé proudy složek dělí při multiplexovaní, na 188 bajtů a tyto fragmenty nazývá pakety. Každá složka má ve svých paketech jednoznačné označení PID – z anglického Packet IDentifier.

Základní struktura paketu vypadá následovně:



Obrázek 4: Struktura TS paketu: levá část - základní struktura paketu s uvedením počtu bitů, které zabírají jednotlivé položky, pravá volitelná část – přizpůsobovací pole

Paket obsahuje minimálně 4 bajty hlaviček, na užitečná data tak zbývá maximálně 184 bajtů (1472 bitů). Pokud je potřeba, mohou být přidány další hlavičky v tzv. přizpůsobovacím poli (adaptation field).

Vysvětlím uvedené položky z paketu:

sync_byte

- bajt sloužící k synchronizaci, každý paket vždy začíná hexadecimální hodnotou 0x47

transport_error_indicator

- pokud zaznamená demodulátor chybu při přenosu, nastaví tento bit na 1

payload_unit_start_indicator

- tento bit při stavu 1 indikuje skutečnost, že v tomto paketu začínají data PES nebo tabulky (popisováno dále)

transport_priority

- umožňuje definovat pakety se zvýšenou prioritou

PID

- nabývá hodnot 0-4095 (13 bitů), označení paketů stejného proudu dat

transport_scrambling_control

- využívá šifrování vysílání

adaptation_field_control

- určuje, zda je v paketu přítomna adaptation field a zda paket nese užitečná data (payload)

continuity_counter (čítač souvislosti)

- 4bitový čítač (hodnoty 0 – 15), který se inkrementuje pro každý paket stejného pidu, každý pid má tedy svůj vlastní nezávislý čítač, výjimku tvoří případ, kdy adaptation_field_control nabývá binárních hodnot 00 a 10 (užitečná data nejsou v paketu přítomna)

adaptation_field (přizpůsobovací pole)

- skupina volitelných hlaviček, které jsou přítomny v případě, že adaptation_field_control nabývá binární hodnoty 10 nebo 11

payload

- výplň, samotná užitečná multiplexovaná data

Přizpůsobovací pole může nabývat ještě větších rozměrů, v závislosti na příznacích PCR_flag, OPCR_flag, splicing_point_flag, transport_private_data_flag a adaptation_field_extension_flag jsou na pozici optional_fields doplněny další hlavičky. Mezi nejdůležitější patří hlavička PCR, která nese dva čítače, prvních 33 bitů tvoří čítač inkrementovaný ve frekvenci 90 kHz, druhá část tvoří 9tibitový čítač inkrementovaný ve frekvenci 27 MHz. Tyto čítače slouží pro synchronizaci audio a video složky.

Výplň nese obvykle dva základní druhy informací – tabulky a PES (Packetized Elementary Stream). Zatímco první z nich slouží spíše pro kontrolní informace, popisovače a doplňkové služby, druhý nese především audio a video data.

2.2.2 Tabulky

Tabulky, jako jeden z typů informací nesených ve výplni TS paketu, zprostředkovávají hned několik méně či více klíčových funkcionalit v rámci celého vysílání. Obsah tabulky nesou tzv. sekce, proto se pojem tabulky a sekce často zaměňuje. Jejich význam vysvětlím u výčtu konkrétních typů, s popisem těch nejdůležitějších:

PAT (Program Association Table)

- seznam programů, které jsou součástí datového toku

PMT (Program Map Table)

- podrobné informace o jednom programu a jeho složkách (video, audio, teletext…)

EIT (Event Information Table)

- informace o vysílaných pořadech

NIT (Network Information Table)

- data o síti (multiplexu)

CAT (Conditional Access Table)

- data sloužící k popisu použitého systému podmíněného přístupu (šifrování)

BAT (Bouquet Association Table)

RST (Running Status Section)

TDT (Time and Date Table)

TOT (Time Offset Table)

SDT (Service Description Table)

TSDT (Transport Stream Description Table)

Každá z tabulek má svůj specifický formát a unikátní identifikátor (table_id), i když mnohé se strukturou podobají. Samotné tabulky však často přímo nenesou většinu užitečných informací, ale spojují informace, které jsou součástí v nich obsažených deskriptorech (popsaných níže), a přidávají doplňkové data a hlavičky.

2.2.3 PAT

Datový proud MPEG2-TS může obsahovat obecně několik různých programů. Úkolem PAT je vytvořit zjednodušený seznam těchto programů s odkazem na PID, ve kterých lze nalézt podrobnější informace. PAT se vždy nachází v PID 0 a má identifikátor tabulky rovný nule. V datovém proudu se tedy PAT opakuje stále dokola a to pouze v jednom unikátním exempláři. Tato skutečnost může z významné části ovlivnit také dobu, než se uživateli zobrazí zvolený kanál, protože dekodér musí nejdříve načíst obsah této tabulky a to může trvat několik desetin sekundy.

table_id	8
section_syntax_indicator	1
'0'	1
reserved	2
section_length	12
transport_stream_id	16
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
for (i = 0; i < N; i++)	
program_number	16
reserved	3
program_map_PID	13
CRC_32	32

Obrázek 5: Struktura sekce nesoucí PAT tabulku – jednotlivé položky a jejich velikost v bitech

Stěžejní část sekce na obrázku 5 – cyklus s výpisem jednotlivých programů, nese číselné označení programu (program_number) a číslo PIDu, který nese bližší popis programu (PMT). Kolik programů proud obsahuje, je odvozeno z délky sekce (section_length), dekodér tedy čte do doby, dokud nenarazí na poslední čtyři bajty sekce s hodnotou CRC.

2.2.4 PMT

Tabulka PMT slouží pro detailní popis každého programu v datovém proudu. Kolik obsahuje proud programů, tolikrát je obsažena v proudu také různá tabulka PMT s různou hodnotou PID. Dekodér před zobrazením kanálu potřebuje znát také obsah této tabulky, avšak její obsah může mít uloženou ve vyrovnávací paměti, pokud došla dříve než PAT, pokud došla později, tak to obvykle nebude znamenat významné zdržení, proto interval posílání PAT a PMT bývá typicky shodný.

table_id	8
section_syntax_indicator	1
'0'	1
reserved	2
section_length	12
program_number	16
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
reserved	3
PCR_PID	13
reserved	4
program_info_length	12
for (i = 0; i < N; i++)	
descriptor()	
for (i = 0; i < N1; i++)	
stream_type	8
reserved	3
elementary_PID	13
reserved	4
ES_info_length	12
for (j = 0; j < N2; j++)	
descriptor()	
CRC_32	32

Obrázek 6: Struktura sekce nesoucí PMT tabulku – jednotlivé položky a jejich velikost v bitech

Strukturu PMT znázorňuje obrázek 6. Hodnota PCR_PID vybírá PID elementárního proudu, ve kterém se vyskytují hodnoty synchronizačního časovače PCR. Provozovatelé obvykle vybírají PID video složky, protože fragmenty jeho toku se v datovém proudu vyskytují nejčastěji, protože mají největší nárok na přenosovou kapacitu. Program jako celek mohou popisovat deskriptory v cyklu části nazvané Program Info, počet deskriptorů je odvozen od délky části uložené v položce program_info_length. Poté již následuje samotný cyklus popisu jednotlivých elementárních proudů (ES - elemetary stream), každý proud má vyhrazený jeden PID unikátní v rámci celého proudu dat. Cyklus obsahuje také identifikaci typu proudu a volitelně může být proud popsán několika deskriptory. Jeden proud odpovídá např. audio nebo video složce.

2.2.5 Deskriptory

V různých typech sekcí mohou být obsaženy informace stejného typu. Také proto pro uložení těchto informací byla vytvořena celá řada deskriptorů, každý typ sloužící pro specifický druh informace. Každý deskriptor obsahuje hlavičku nesoucí identifikační číslo typu deskriptoru (descriptor_tag) a délku dat v něm obsažených (descriptor_length), takže dekodér v přijímači teoreticky nepotřebuje všechny typy znát a s postupem času mohou být přidány typy nové, které staré přijímače bez jejich podpory prostě ignorují.

2.2.6 PES

Již z anglického názvu Packetized elementary stream (PES) lze vyčíst, že výplň MPEG2-TS paketů tohoto typu bude nést elementární proudy programu, především tedy audio a video složky. Nejedná se však o jediné uplatnění, PES se používá také např. pro zaobalení teletextových dat nebo ECM a EMM dat pro realizaci podmíněného přístupu. Konkrétní význam dat určuje položka stream_id v hlavičce PES paketu.

Struktura PES paketu se odvíjí od položky stream_id a množství volitelných hlaviček, takže uvedu základní strukturu používanou především u audio a video dat, kompletní strukturu lze dohledat v [1, tabulka 2-17].

Mezi hlavní hlavičky PES paketu (obrázek 7) patří:

packet_start_code_prefix

- identifikace začátku paketu, vždy hodnota 0x000001

stream_id

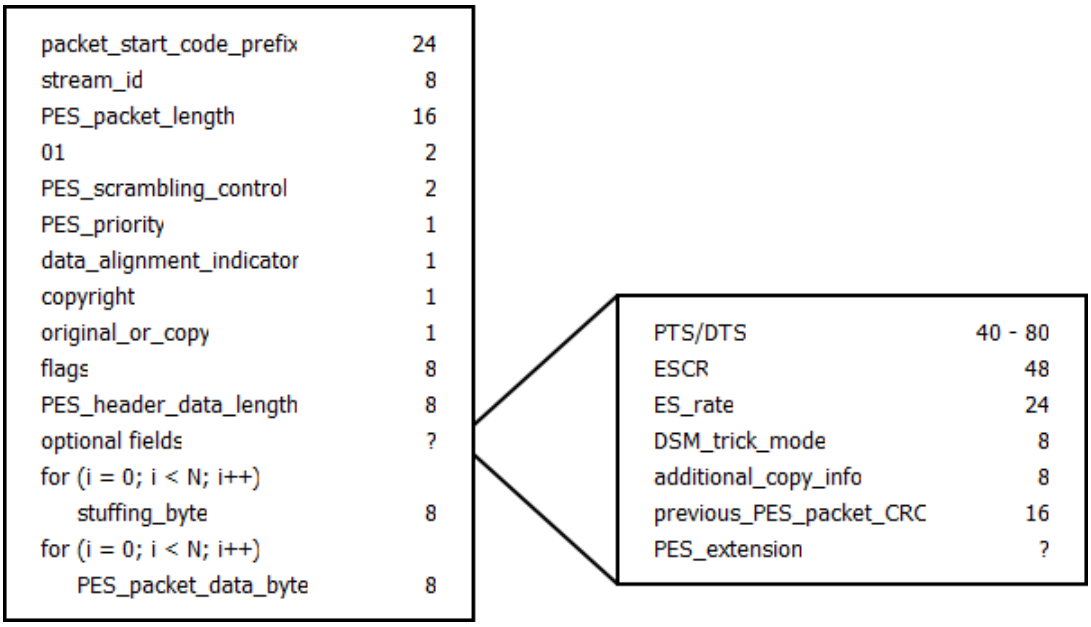
- udává typ nesených dat, celkem možných 256 hodnot, z toho 16 vyhrazených pro definici video kodeku a 32 pro definici audio kodeku, detailněji [1. tabulka 2-18].

PES_packet_length

- udává velikost dat, které obsahuje ve výplni celý PES paket (může se rozkládat přes několik MPEG2-TS paketů), výjimku tvoří video data, které mají délku paketu vždy 0

PES_packet_data_byte

- samotná data, které PES paket nese



Obrázek 7: Struktura PES paketu pro nejběžnější data s uvedením počtu bitů, které zabírají jednotlivé položky

3. IPTV vysílání

Zatím jsem se v textu této práce věnoval do menších či větších detailů některým principům DVB vysílání, avšak jak ukážu, tak se z velké části kryjí s principy živého vysílání po internetovém protokolu (IP).

3.1 Proč TV přes IP

Dalo by se říci, že příchod Internetu znamenal v telekomunikacích impuls ke změně v několika zavedených principech. U tradičních pevných telefonních služeb dominovala technologie přenosu po dvoupárovém metalickém vedení, u televizních služeb převažovaly technologie pozemního nebo satelitního příjmu a později kabelové televize, která však nepřinesla po technologické stránce žádné významné změny. Do tohoto prostředí vznikl Internet a musel si najít způsob, jak se dostat k uživatelům do každé domácnosti a kanceláře. Zpočátku si propůjčil ke svému přežití telefonní linku, později si uživatelé zvláště kvůli dostupné přenosové kapacitě oblíbili připojení přes rozvody kabelové televize, avšak především v posledních letech lze sledovat rozvoj sítí budovaných primárně pro internetové služby. Společně s IP protokolem se jako dominantní vyprofilovaly řešení založené na technologii Ethernet, na páteřních (ale nově i v přístupových) sítích na optickém vlákně, v přístupových sítích na metalickém čtyřpáru, v České republice dosáhly vysoké obliby, kvůli specifickým podmínkám, také sítě bezdrátové.

Se zavedením relativně dostupné a levné časově neomezené internetové přípojky lze sledovat pomalou konvergenci telefonních a televizních služeb právě do IP prostředí. Vznikají tak možná až podivné kombinace, kdy se pro přenos TV použije IP a ten následně putuje přes telefonní linku nebo se pro přenos telefonních hovorů použije IP přes kabelovou televizi. Pokud už TV provozovat přes IP, má dnes smysl použít pouze přístupové sítě na bázi optického přenosu FTTx. Výstavba těchto sítí u nás zažívá živelný rozvoj až v posledních letech, takže ruku v ruce je i IPTV celkem mladá a rozvíjející se služba. Někdo by mohl namítnout, že v roce 2011 je největším poskytovatelem IPTV společnost, která tuto službu nabízí na telefonním vedení, a nejdostupnější rychlé a ještě cenově přijatelné připojení k Internetu nabízí jedna společnost přes kabelové (koaxiální) rozvody, avšak nejen dle mého názoru bude budoucnost patřit přenos těchto tří služeb (internet, telefon, televize) právě přes Ethernet a IP protokol, i když to tak asi hlavně kvůli roztříštěnosti provozovatelů internetového připojení nevypadá.

3.2 Vztah k DVB

A proč se tedy živé televizní vysílání přes IPTV tak podobá DVB? Je to jednoduché, jednotlivé MPEG2-TS pakety programu u vysilatele nesměřují do multiplexeru a modulátoru, ale jsou ve většině případů zabaleny přímo do UDP datagramů a poslány do IP sítě ve formě tzv. multicastového přenosu. Děje se tak tedy ve většině případů, někteří vysilatelé z různých důvodů volí ještě jiné metody enkapsulace, avšak u živého vysílání se tento jednoduchý přenos paketů přes UDP nejčastější. Multicastový přenos umožňuje to, aby na páteřních linkách byl každý z kanálu přenášen ideálně maximálně jedenkrát nezávisle na počtu sledujících diváků. Každému jednomu programu je přidělena jedna multicast skupina, protože přijímač se dokáže přihlásit k jedné skupině jako celek a nedokáže rozlišovat více programů v rámci jedné skupiny, jeden tok programu obsahuje tedy právě jednu PAT a typicky právě jednu PMT, jednu video složku v případě TV kanálu (žádnou v případě rádiového programu), alespoň jednu audio složku, volitelně také teletext a další služby.

3.3 Systémové problémy

Stejně jako v případě přenosu IP přes telefonní linky a podobné, tak i Ethernet a IP nebyly navrženy s tím, že by přes ně bylo směřováno televizní vysílání, u kterého jsou kladeny velké nároky na kvalitu a spolehlivost. Z toho vyplývá několik výzev a problému, s kterými se IPTV musí vypořádat.

Právě proto, že je při IPTV vysílání vyřazen modulátor v pojetí tak, jak ho chápe DVB, a modulaci a demodulaci provádí Ethernet zařízení, které rozpozná díky kontrolnímu součtu pouze chybu v přenesených datech, nelze počítat s tím, že by šlo částečně poškozený datový tok rekonstruovat tak, jak je to běžné u DVB vysílání, kde modulátor přidává opravný kód (FEC). V Ethernet prostředí bývá navíc běžnější jev, že rámec nedojde vůbec, než že dojde poškozený, protože by muselo dojít k závadě přímo na přípojce u uživatele, protože aktivní prvky po cestě vadné rámce (a tím pádem UDP diagramy) zpravidla zahazují.

Data IPTV a běžného Internetu spolu obvykle koexistují na společných datových linkách, zcela oddělenou infrastrukturu pro dvě služby nelze z ekonomických důvodů zajistit. Běžná internetová data jsou často přenášena ve shlucích a zabírají přenosovou kapacitu nárazově, takže se stává, že jsou páteřní nebo dokonce i koncové linky na zlomky sekund přetíženy. Tento problém řeší implementace tzv. kvality služeb (QoS) do sítě, kdy lze jedny data upřednostnit před druhými. Aktivní prvky s podporou této technologie bývají však často dražší a ne vždy je jejich implementace bezchybná, takže pokud tento mechanismus nefunguje v pořádku, může docházet k zahazování rámců s TV daty a tím pádem k poruchám v příjmu.

Princip multicast - přenos televizních dat více příjemcům, je navržen již od počátku Ethernetu a IP, avšak metody směrování jejich toku v závislosti na přihlášených účastnících procházely v čase změnami. Různé implementace lze nalézt až v aktivních prvcích ve vyšších cenových relacích. Bez nich prakticky nelze větší míře multicast provozovat, protože pak dochází k šíření nevyžádaných dat do částí sítě, kde o ně nikdo aktuálně nemá zájem, což vede ke zbytečnému zatížení nebo

dokonce jejímu přetížení a znefunkčnění. Při multicast přenosu nelze použít spolehlivý protokol TCP, aby byla zajištěna souvislost a kompletnost celého toku.

Svou roli ve spolehlivosti hraje počet aktivních prvků po trase mezi vysilatelem a příjemcem. V případě satelitního vysílání, když dojde ke správnému vyslání dat ze satelitu, tak již žádný aktivní prvek po trase k Vaší parabole není. V případě přenosu kabelové televize po cestě naleznete maximálně nějaké zesilovače a podobné aktivní prvky, které mohou být i centrálně napájeny. U přenosu IPTV bývá situace často opačná, často po cestě k zákazníkovi naleznete řadu aktivních prvků, které znamenají potencionální poruchové místo a mnohé z nich nemusí být ani zálohovány proti výpadku elektrické energie. Proti výpadku zařízení nebo jejich napájení má Ethernet i IP několik doprovodných technologií, aby byla zajištěna funkčnost alternativními trasami, avšak doba konvergence není nekonečně malá a proto každá taková změna vede k poruchám příjmu, navíc budování záložních tras a prvků nese své finanční náklady. Potřebu aktivních prvků a napájení elektrickým proudem významně odbourává technologie pasivních optických sítí (PON), k výstavbě těchto sítí dochází i v České republice, avšak díky konkurenčním tlakům, vysokým počátečním investicím a nákladům na připojení klienta je takto připojeno maximálně několik málo desítek tisíc uživatelů.

4. Pomocné knihovny

Už při návrhu řešení úkolů ze zadání práce mi bylo zřejmé, že bude zapotřebí vytvořit programové knihovny, které zabezpečí opakované funkce a zjednoduší vývoj a přehlednost zdrojových kódů výsledných aplikací.

Vytvořil jsem dvě základní knihovny – libdvb a libnet. První z nich ulehčuje práci s DVB daty (MPEG2-TS) a druhá pomáhá především při síťových operací v IP prostředí.

4.1 Knihovna libdvb pro práci s DVB daty

Knihovna libdvb nabízí možnosti, jak pracovat s daty ve formátu MPEG2-TS, ať se jedná o jejich interpretaci, dekódování, transformaci, kódování nebo jiné činnosti. Při jejím vytváření jsem nekladl důraz na její kompletní implementaci ve všech aspektech, ale aby zajišťovala požadované funkce pro vytvářené aplikace.

4.1.1 Dekodér

Stěžejní části knihovny libdvb tvoří dekódovací část, která z multiplexovaného toku, toku jednoho programu nebo obecně jakéhokoliv korektního toku MPEG2-TS dat dokáže separovat relevantní požadovaná data. Jistá podobnost s veřejně dostupnou knihovnou libdvbpsi, která plní podobné funkce, je možná, avšak svou knihovnu jsem vytvářel pro své specifické požadavky, než abych se zabýval přetvářením řešení, které si klade trochu jiné cíle.

Do dekodéru vstupují obecně jakákoliv MPEG2-TS data, přes zpětná volání vystupují volitelná data na základě parametrů, výstupy mohou být:

- pakety zadaného PID
- pakety zadeného programu (PNR)
- pakety daného typu (odpovídající položce stream_type z PMT)
- deskriptory zvoleného typu
- sekce daného typu (odpovídající položka table_id)
- PES se zvoleným stream_id (pokud má PES nenulovou velikost)
- nepoškozená data
- kombinace předchozích parametrů (ty, které jsou kombinovatelné)

Lze takto efektivně za pomocí dobře specifikovaného parametru vyžádat data z komplexního datového toku. Dále poskytuje dekodér několik doprovodných funkcí:

- informace o změně PAT a PMT
- analýza a statistika poškozených dat (transport error indicator, čítač souvislosti, CRC32)
- možnost paralelního zpracování náhrady poškozených dat
- filtrování nepožadovaných dat (na které nevede žádný filtr) pro zlepšení výkonu
- dekódování a interpretace vybraných sekcí a deskriptorů

Význam některých funkcí popisuji dále při řešení konkrétních problémů. Dekodér ukládá příchozí data do kruhové vyrovnávací paměti, do které směřuje několik ukazatelů, podle toho, v jaké úrovni zpracování data aktuálně jsou (podrobněji v 8.3.3 a na obrázku 17).

4.1.2 Enkodér

Implementoval jsem velmi jednoduchý dekodér MPEG2-TS dat, nebo by se dalo také říci multiplexer. Enkodéru se předloží PMT požadovaného datového toku, poté se již předkládají pakety libovolně komplexního datového toku, enkodér z něho vyfiltruje pakety s PID, které popisují předložené PMT a v pravidelných intervalech generuje pakety s PAT a PMT.

Tato část knihovny se uplatní při skládání (multiplexování) několika elementárních toků do jednoho programu, slučování toků více programů, apod.

4.1.3 JIT

Mezi důležitou část knihovny řadím také funkcionalitu tzv. just-in-time (JIT), kdy se knihovna snaží rekonstruovat správně časování toku paketů MPEG2-TS tak, jak byl vysílán. Při časově nerovnoměrného přenosu paketů sítí (tzv. jitter), z důvodu zpracování nebo při jiném uváznutí ve vyrovnávací paměti může totiž dojít k situaci, že by přijímač IPTV začal přehrávat datový tok, avšak díky opožděnému přijetí dat by došlo k podtečení vyrovnávací paměti a přehrávání by musel znovu zastavit. JIT se proto postará, že vytvoří jistý „polštář“ ve formě vyrovnávací paměti a za pomoci analýzy časovače PCR v datovém toku dávkuje pakety přesně ve správný okamžik, pokud sám zjistí, že mu vyrovnávací paměť podtekla, provede zvýšení jejího minimálního zaplnění.

Z principu fungovaní nelze tuto funkci použít u toku více než jednoho programu nebo obecně toku, který obsahuje více paketů nesoucích PCR. Také je zapotřebí počítat s tím, že mezi dvěma po sobě jdoucími pakety s PCR nelze již jemnější časování provádět, proto v případě, kdy dojde na čas, kdy se může aktuální paket s PCR na vrcholu vyrovnávací paměti odeslat, budou ihned odeslány také všechny následující pakety, které PCR neobsahují.

JIT může pracovat ve dvou modech – blokovaném a neblokovaném. U prvního jmenovaného dochází k blokování čtení dat ze vstupu, dokud nejsou předešlá data zpracována a nemá dojít k odeslání balíku paketů s dalším PCR, u druhého způsobu jsou vstupní data vždy uložena do vyrovnávací paměti a z ní uvolňována průběžně v čase podle PCR. Blokované zpracovaní lze využít například pro přehrávání záznamu vysílání z pevného disku, neblokovaný přístup se uplatní hlavně v živém vysílání.

4.1.4 Ostatní

Knihovna poskytuje ještě několik dalších podpůrných funkcí:

- kontrola CRC u sekcí
- paketizér pro extrakci jednotlivých paketů z datového toku

4.2 Knihovna libnet pro práci se sítí

Při vysílání prostřednictvím IPTV hrají důležitou roli síťové operace. Protože se mnohé úkoly ve vytvářených programech podobaly a ve zvoleném jazyce C nemusí být vždy dobře čitelné, co je konstrukcemi pro práci se síti chápáno, vytvořil jsem k tomuto účelu pomocnou knihovnu libnet, která umožňuje především zjednodušený přístup ke zdrojům dat a následnou distribuci do zpět sítě.

4.2.1 TCP funkce

Tato část knihovny se zabývá práci s TCP spojeními. Zkráceně lze nabízené funkce popsat takto:

- připojení na zadaného hostitele a port blokovaně či neblokovaně
- připojení na zadaného hostitele a port neblokovaně s časovým limitem
- vytvoření TCP serveru na zadaném portu blokovaně či neblokovaně
- neblokovaný zápis do spojení ošetřený časovým limitem
- neblokovaný zápis do spojení ošetřený časovým limitem, příchozí data jsou zahazována a hlídají se výjimky a odpojení klienta
- načítání dat ze zadaného spojení po jednom bajtu, pokud není načten požadovaný řetězec
- neblokované čtení ze spojení ošetřené časovým limitem

4.2.2 TCP server

Knihovna umožňuje vytvoření TCP serveru a vlákna, které zpracovává příchozí připojení. O novém připojení je aplikace informována přes zpětné volání a je už na ní, jak s ním naloží.

Protože jsem tuto funkcionalitu použil opakovaně k vytvoření primitivního HTTP serveru, jsou obsaženy také funkce pro načítání HTTP hlaviček a zpracování GET parametrů.

4.2.3 TCP datový server (TDS)

Zatímco předchozí TCP server slouží pro obousměrnou různorodou komunikaci klient – server, TCP datový server jsem vytvořil za účelem jednosměrného posílání stejných dat více adresátům, kteří se na daný server připojí.

TDS zpracovává příchozí spojení také v samostatném vlákně, o připojení klienta informuje aplikaci prostřednictvím zpětného volání, ta může například zpracovat nějaké požadavky od klienta nebo přečíst HTTP hlavičky a pak už je spojení pouze v módu pro zápis, kdy jsou všechna příchozí data zahazována. Knihovna se stará o duplikaci posílaných dat do jednotlivých spojení, každé spojení má vyhrazeno samostatné vlákno s neblokovaným zápisem, případně sama odpojuje neaktivní klienty.

4.2.4 Přístup ke zdrojům

Ať soubory, UDP,TCP nebo specificky HTTP spojení v systému reprezentuje deskriptor s jednotným přístupem ke čtení a zápisu, proto jsem vytvořil jednotný způsob, jak vytvářet deskriptor k těmto zdrojům.

Zdroj může být popsán např. těmito řetězci:

- prázdný řetězec nebo „stdin“
  - čtení ze standardního vstupu
- http://1.2.3.4/test.php
  - vytvoření TCP spojení na daný server a zaslání HTTP GET hlaviček
- udp://@233.33.33.33:1234/
  - vytvoření přijímače dat z dané multicast skupiny na daném portu včetně zasílání IGMP zpráv pro registraci do skupiny
- komp://1.2.3.4:1234/
  - pro zdroj dat použije připojení na tzv. kompenzační server (popisovaný později)
- cokoliv ostatní
  - název souboru, který se otevře

Pro práci s vytvořeným deskriptorem mohou být použity identické funkce z části TCP server a klient, avšak vytvořil jsem také speciální dvě funkce, které zabezpečují případné opětovné připojení zdroje v případě chyby:

- blokované čtení z deskriptoru, pokud dojde k chybě, dojde k opakovanému připojení
- blokované čtení z deskriptoru, pokud dojde k chybě, dojde k opakovanému připojení a aplikace je o tom informována prostřednictvím zpětného volání

Opětovné připojení používá specifický algoritmus, který zajišťuje, že první pokusy jsou prováděny bezprostředně, další pokusy se zvyšujícím se intervalem až do pěti sekund.

Funkce jsem implementoval jak pro blokované, tak i neblokované připojení a čtení, u neblokované varianty lze čas připojení a čtení omezit časovým limitem.

4.2.5 Ostatní

Součástí knihovny jsou některé další funkce pro vytváření multicast klienta a funkce pro zajištění kompatibility s operačním systémem Windows. Díky potřebě provozu pouze na systému Linux a značným rozdílům ve funkčnosti některých záležitostí pod systémem Windows nebyla knihovna dlouhodobě udržována pro možnost kompilace a korektního běhu na tomto operačním systému.

5. Dekodér elektronického programového průvodce

Jeden z velkých argumentů pro přechod na digitální televizní vysílání, který mohli diváci slyšet, byla existence elektronického programového průvodce (EPG). Tato funkce jim přímo na obrazovce televizoru ukáže, jaký pořad hraje, jeho popis, typ pořadu, ale mohou si vyhledat také informace o budoucích pořadech až na několik dní dopředu. Jisté standardizované formy EPG lze nalézt sice už v analogovém vysílání, ale praktického masivního nasazení se nedočkaly.

5.1 Struktura dat EPG

Nedá se říci, že bychom mohli v datovém toku najít EPG data připravené na jednom místě, jenom je interpretovali a zobrazili uživateli. Jedná se spíše o kombinaci více druhů dílčích informací, které tvoří kompletní informační servis pro uživatele.

5.1.1 Nosné tabulky a deskriptory

Většinu relevantních informací lze nalézt v deskriptorech, které nese tabulka Event informational table (EIT, obrázek 8). Sekce s EIT mají vyhrazen vždy PID 18.

Kompletní seznam možných deskriptorů v EIT popisuje [1, tabulka 12], avšak s mnohými se v praxi člověk nesetká nebo neplní pro uživatele významnější informační roli, mezi klíčové patří:

- *short event descriptor*
  - umožňuje přiřadit k události název a krátký popis (struktura na obrázku 9)
- *extended event descriptor*
  - podrobný popis události
  - velikost deskriptoru je sice omezena rozsahem hodnoty descriptor_length na 255 bajtů, avšak deskriptor umožňuje rozšíření až přes 16 následujících deskriptorů za pomocí položek descriptor_number a last_descriptor_number, lze tak vytvořit velmi obsáhlý popis
- *component descriptor*
  - popis složek události – standardní nebo vysoké rozlišení, poměr a frekvence obrazu, použitý kodek pro audio a video, existence titulků, typ zvuku (mono, stereo, prostorový, vícejazyčná stopa), apod.
- *content descriptor*
  - definice typu obsahu – obecný (např. sport, film, novinky, dětský pořad, hudební pořad) a podrobný (hudební pořad s jazzem, klasickou, rockovou hudbou, apod.)
- *parental rating descriptor*
  - pro jaké věkové skupiny je pořad vhodný v různých zemích (v rozsahu 3 až 18 let)

table_id	8
section_syntax_indicator	1
'0'	1
reserved	2
section_length	12
service_id	16
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
transport_stream_id	16
original_network_id	16
segment_last_section_number	8
last_table_id	8
for (i = 0; i < N1; i++)	
event_id	16
start_time	40
duration	24
running_status	3
free_CA_mode	1
descriptors_loop_length	12
for (j = 0; j < N2; j++)	
descriptor()	
CRC_32	32

Obrázek 8: Struktura sekce nesoucí EIT tabulku –jednotlivé položky a jejich velikost v bitech

descriptor_tag	8
descriptor_length	8
ISO_639_language_code	24
event_name_length	8
for (i=0;i<event_name_length;i++)	
event_name_char	8
text_length	8
for (i=0;i<text_length;i++)	
text_char	8

Obrázek 9: Short event descriptor – deskriptor nesoucí název a krátký popis události (v popsaném jazyce)

5.1.2 Vazby mezi informacemi

Položka service_id v EIT (obrázek 8) odpovídá číslu programu (program_number, PNR) z PAT a PMT, takže data z aktuální tabulky se budou vztahovat pouze k jednomu určenému programu. EIT dále obsahuje cyklus událostí (pořadů). K programu tedy může být přiřazeno několik různých událostí pod unikátním event_id, u kterých je uveden čas jejich začátku (ve formátu modifikovaného juliánského kalendáře MJD v koordinovaném světovém čase UTC), délka trvání, informace o běhu události a zda podléhá systému podmíněného přístupu, další podrobnosti popisují doprovodné deskriptory. Velikost sekce je omezena maximální hodnotou položky service_id, takže všechny události programu v rámci ní nelze obvykle popsat, proto se dělí do několika navazujících sekcí, jejich celkový počet a aktuální index je uložen v položkách section_number a last_section_number.

5.2 Implementace

Vývoj EPG dekodéru i vzhledem k předchozí zkušenosti s touto problematikou neznamenal nějaký zásadní problém, díky vytvořeným knihovnám se nejedná ani o příliš velkou aplikaci, podstatnou část zdrojového kódu zabírá kód pro uchovávání a export zjištěných dat.

5.2.1 Kódování speciálních znaků

Z pohledu standardu [1] lze chápat speciálním znakem také písmena s českou diakritikou, proto jsem byl nucen řešit problém s převodem vnitřního formátu znaků do některého z obecně používanějšího formátu, zvolil jsem UTF-8.

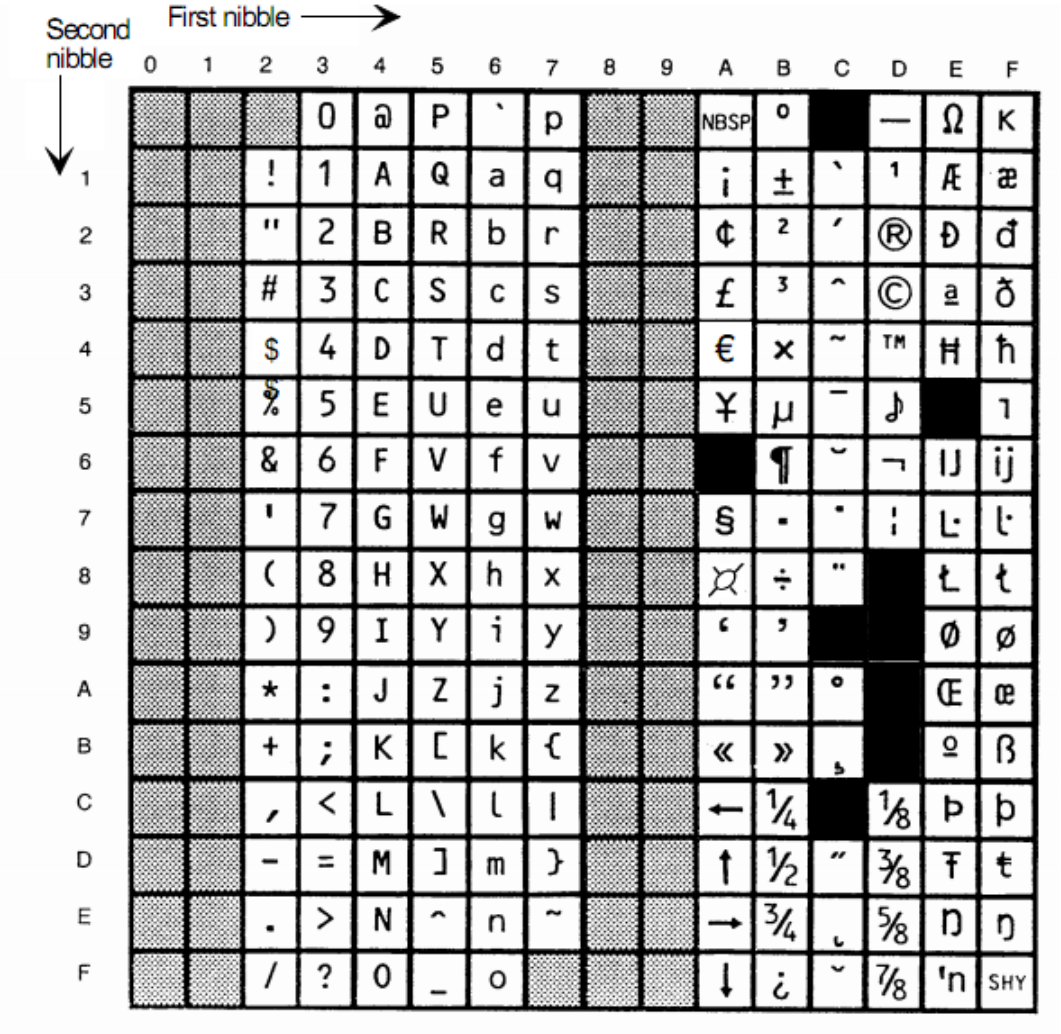
Všechny textové hodnoty, na které se vnitřní formát znaků vztahuje, používají pro interpretaci tato pravidla:

- text se zpracovává po jednom bajtu
- první bajt textu určuje, jaká sada znaků se použije, pokud nabývá hodnoty 0x20 a vyšší, použije se sada latinské abecedy, hodnoty menší lze podle standardu použít pro volbu sady jiné, např. kombinované latinská/cyrilice, latinská/arabská, latinská/řecká, latinská/hebrejská, apod. (vybraná tabulka je součástí normy [1])

Pro základní tabulku latinských znaků platí pravidla popsaná na obrázku 10:

- většina pozic s hodnotami znaků 0x20 až 0x7F je přímo interpretovatelných (přímý převod do UTF-8, binárně kompatibilní)
- znaky na pozicích 0xC0 až 0xCF slouží jako diakritická znaménka, které tvoří kompletní znak až dohromady s následujícím bajtem (tedy např. čárka na pozici 0xC2 a následně znak A na pozici 0x41 lze interpretovat jako Á)
- zbytek pozic v tabulce tvoří neobvyklé nebo vyhrazené znaky

V EPG dekodéru jsem implementoval interpretaci pomocí tabulky latinských znaků, vzhledem k tomu, že nelze jednoduše strojově převádět převod nepřímo interpretovatelných znaků do UTF-8, zahrnul jsem vybrané znaky s diakritickými znaménky, které jsou běžné v regionu střední Evropy s tím, že není problém do převodní tabulky doplnit převody další.



NOTE 1: The SPACE character is located in position 20h of the code table.
NOTE 2: NBSP = no-break space.
NOTE 3: SHY = soft hyphen.
NOTE 4: This table is a superset of ISO/IEC 6937 [22] with addition of the Euro symbol.
NOTE 5: All characters in column C are non-spacing characters (diacritical marks).

Obrázek 10: Tabulka latinských znaků [1, obrázek A.1]

5.2.2 Sběr dat

Výsledná konzolová aplikace přijímá zadání přes parametry příkazové řádky. Povinný parametr -z definuje zdroj dat, který se použije pro získávání EPG dat, může se jednat o jakýkoliv zdroj podporovaný knihovnou libnet, nejlépe datový tok z živého vysílání DVB multiplexu obsahující pouze data z PID 0 a 18.

5.2.3 Vyhodnocování

Aplikace používá pro extrakci deskriptorů z datového toku dekodér knihovny libdvb, pro každý deskriptor zaregistruje příslušný filtr, do dekodéru posílá celý datový tok ze zdroje a přes zpětné volání dostává už konkrétní vyseparované deskriptory, žádná další logika ohledně datového toku není potřeba.

EPG dekodér deskriptory dekóduje a předá do interní databáze událostí, která zajistí spojení všech relevantních údajů k odpovídajícím událostem. Databáze se stará také o odstraňování záznamů o již neplatných událostech.

Kromě uvedených EIT deskriptorů dokáže navíc dekodér zpracovávat obsah deskriptoru Service descriptor z tabulky Service description table (SDT), aby bylo možno jednotlivé programy identifikovat podle názvu (místo identifikátoru service_id) a jejich provozovatele.

Během implementace a testování jsem přišel na nutnost porovnávání položky transport_stream_id v tabulkách s hodnotou transport_stream_id v PAT, protože někteří DVB vysilatelé vysílají EPG data k programům, které nejsou vůbec součástí multiplexu.

5.2.4 Export MySQL

Dekodér umožňuje volitelný online export všech získaných dat do MySQL databáze. Tento způsob umožňuje velmi jednoduché propojení do stávajících nebo nových informačních systémů pracujících s touto relační databází.

Při použití MySQL exportu je potřeba počítat s jeho vyšší výpočetní náročností především v době ihned po startu, protože dochází k novému naplnění databáze velkým objemem nových informací z datového toku. Z principu fungování vyžaduje každá instance EPG dekodéru samostatnou databázi v MySQL s pracovními tabulkami, při každém spuštění dochází k jejímu kompletnímu vyprázdnění.

5.2.5 Export HTTP

Na základě funkčního MySQL exportu jsem vytvořil PHP skript, který sloužil pro následný export dat v XML, nicméně jsem zjistil, že výsledné řešení co do výpočetní náročnosti neodpovídá požadovaným nárokům. Proto jsem se rozhodl, že vytvořím ještě jeden typ exportu, který bude generovat přímo EPG dekodér, nebude tak docházet k vysoké režii databáze, skriptovacího jazyka webové aplikace a webového serveru, a bude rovněž jednoduše implementovatelný do jiných systému, avšak nespotřebuje tolik systémových prostředků. Pro splnění těchto požadavků jsem musel opustit myšlenku standardizovaného formátu XML a vytvořit proprietární a značně zjednodušenou formu interpretace exportovaných dat. Pro přenos dat jsem vybral protokol HTTP, EPG dekodér tedy používá implementaci jednoduchého HTTP serveru za pomocí knihovny libnet.

Zvolil jsem formát, kdy každý řádek (zalomení řádku jako oddělovač) znamená samostatnou událost, skupiny údajů odděluje znak „|”, položky ve skupině údajů znak „;“, v hlubší hierarchii „*“. Součástí první skupiny jsou povinné údaje: identifikace programu (service_id), identifikace události (event_id), čas začátku události (v unix time) a jeho délka. Následuje skupina pěti položek, která každá znamená binární značku (v textové podobě 0 nebo 1), zda je každá z následujících pěti

skupin vyplněna daty. Do těchto pěti skupin patří údaje získané z deskriptorů Short event descriptor, Extended event descriptor, Component descriptor, Content descriptor a Parental rating descriptor.

Jednoduchý záznam o události může tady vypadat takto:

```
8001;45994;1297720800;1800|1;0;0;0;0|cze;Noční události;|||||
```

Výpis popisuje informace o pořadu s názvem „Noční události“ (česky), tento pořad začne na stanici s identifikátorem 8001 ve 22 hodin 14. února 2011 koordinovaného světového času (unix time 1297720800) a bude trvat 30 minut (1800 sekund).

Pokud vysilatel nabízí rozšířené informace, může záznam vypadat složitěji:

```
257;44020;1297554300;1800|1;1;1;1;1|cze;Californication II, A žili šťastně...;12/12 Jak překonat krizi středního věku - návod pro otrlé. David Duchovny v hlavní roli amerického komediálního seriálu (2008). Dále hrají: N. McElhoneová, M. Martinová, M. Zimaová, E. Handler|cze;Hank dokončí Ashbyho memoáry. Sonja porodí dítě, které určitě není Hankovo. Charlie dostane práci v prodejně BMW a poznává, že život s Daisy nebude žádné peříčko. Návrat k Marcy se ale zdá být nemožným. Karen přijme pracovní nabídku z New Yorku a všichni se chystají na východ. Když se Becca dá znovu dohromady s Damienem, chce zůstat v L.A. Hank se rozhodne zůstat s ní.|cze*1*3*1*;cze*2*3*2*;cze*3*1*4*|1*8|CZE*18|
```

Popis této události již obsahuje podrobné textové údaje, informaci o tom, že je vhodný v ČR od 18ti let a podle tabulek ve standardu [1, tabulka 16 a 18] bychom dokázali zjistit, že se jedná o film nebo drama pro dospělé, bude se vysílat ve standardním širokoúhlém rozlišení, stereo zvukem a se skrytými titulky na teletextu.

Ve výchozím stavu dekodér exportuje pouze základní informace - údaje získané z deskriptorů Component descriptor, Content descriptor a Parental rating descriptor nevrací ani v případě, že jsou od vysilatele k dispozici, avšak toto chování lze upravit podle potřeb žadatele za pomocí HTTP GET parametrů component, content a parental, volitelně lze vypnout posílání údajů z Extended event descriptor za pomocí parametru extended. Pro zapnutí musí parameter obsahovat hodnotu 1, pro vypnutí 0. Pokud jsou vyžadovány informace o událostech pouze z jednoho programu, lze aplikovat filtrování za pomocí parametru service_id s hodnotou čísla požadovaného programu.

5.3 Zhodnocení

Podařilo se mi vytvořit funkční aplikaci pro extrakci informací elektronického programového průvodce z datového toku DVB bez toho, aniž by byla potřeba přímá komunikace s jakýmkoliv hardware. Implementované řešení tak lze zasadit do již funkční infrastruktury vysilatele nebo zpracovatele DVB dat bez dalších nákladů a získat rychle a výkonově efektivně užitečná data pro uživatelské aplikace v IPTV set-top-boxu nebo do jiných komunikačních kanálů a služeb.

EPG dekodér lze jistě rozšířit o další funkce nebo vylepšit funkce stávající, ať se jedná o exporty dat, komplexnější podpora znakových sad různých národních jazyků nebo další záležitosti.

6. Dekodér teletextu

Teletext jistě již dnes nepatří mezi tolik populární služby spojené s televizním vysíláním, avšak i v digitalizovaném vysílání přežívá a dále ho vysilatelé udržují a používají. Vzhledem k tomu, že platná legislativa v ČR předkládá provozovatelům převzatého televizního vysílání povinnost šířit vysílání v kompletní a nezměněné formě včetně doprovodných služeb jako je teletext, nelze tuto službu zcela ignorovat a přístup k nim uživatelům odepřít. Mnohá hotová řešení pro dekódování teletextového proudu lze najít, avšak hotové aplikace typicky předpokládají přímou komunikaci s televizní kartou, což jistě není zcela efektivní po nákladové stránce, ale také na správu, údržbu. Rozhodl jsem se tedy vytvořit vlastní implementaci, která by stejně jako u EPG dekodéru znamenala co nejmenší náklady a výkonově efektivně umožnila interpretaci teletextových dat extrahovaných z proudu MPEG2-TS dat.

6.1 Struktura dat teletextu

Základní stavební jednotkou teletextu v analogovém vysílání je paket o velikosti 45 bajtů. Zatímco v analogovém vysílání jsou pakety modulovány na místě nezobrazovaných řádků, v případě DVB takový funkční ekvivalent už nenajdeme. V digitálním vysílání však nebyl teletext nijak významně přepracován, pouze došlo k následujícím úpravám, které popisuje standard [3]:

- odstranění úvodních dvou bajtů (Clock run-in) pro synchronizaci dekodéru
- každému paketu předchází informace o paritě a identifikace neviditelného řádku (VBI), do kterého daný paket patří v analogovém vysílání, tyto hlavičky a zkrácený paket se nazývá datové pole
- doplní se informace o velikosti datového pole (u teletextu vždy 44 bajtů) a identifikátor typu datové jednotky
- datové pole a hlavičky se mohou zřetězit do větší skupiny
- skupina se opatří identifikátorem dat (pro teletext 0x10)
- všechny uvedené data se zabalí do PES a vysílají v samostatném elementárním proudu programu

6.1.1 Kódování

Existují celkem tři různé kódování, které se v souvislosti s teletextem používají:

- lichá parita
    - kódování sedmi bitů za pomocí osmi bitů
    - může být rozpoznána chyba v jednom bitu
- Hammingovo kódování 8/4
    - kódování čtyř bitů za pomocí osmi bitů
    - jeden vadný bit může být identifikován a opraven a dva vadné bity mohou být identifikovány
- Hammingovo kódování 24/18
    - kódování 18ti bitů za pomocí 24 bitů
    - stejně jako u kódování 8/4 jeden bit může být identifikován a opraven a detekovány mohou být maximálně dva vadné bity

Dekodér opravné a detekční bity nemusí používat, stačí, když je vypustí a zbytek dat dále zpracovává.

6.1.2 Struktura paketu

Vzhledem k tomu, že struktura původního 45bajtového paketu zůstala z analogového vysílání až na počáteční dva bajty naprosto stejná, mohl jsem vyjít ze starého standardu ETS 300 706 [4]. Počáteční hlavička paketu Framing code slouží pro synchronizaci dekodéru a identifikaci začátku paketu a obsahuje vždy stejnou konstantu 0xE4. Následují dva bajty kódované za pomocí Hammingova kódování 8/4, které po dekódování nesou 3bitovou hodnotu Magazine a 5bitovou hodnotu Packet number (označovaná jako Y). V závislosti na hodnotě Packet number se rozlišují tři typy paketů:

- hlavička teletextové stránky (nultý řádek, Y = 0)
- normální paket s přímo zobrazitelnými daty (Y = 1 až 25, Y odpovídá číslu řádky)
- pakety s přímo nezobrazitelnými daty (Y = 26 až 31)

V normálním paketu po zpracování hlaviček zbývá 40 bajtů, každý odpovídá jednomu sloupci teletextové stránky na jednom řádku. Vzhledem k tomu, že bajt obsahuje jeden bit jako lichou paritu, zbývá pouze 7 bitů, které mohou reprezentovat zobrazený znak, ve skutečnosti je však ještě dalších 32 hodnot vyhrazeno pro formátovací symboly.

Paket s hlavičkou teletextové stránky obsahuje ještě dalších 8 bajtů, které nesou číslo stránky a podstránky a některé další pomocné značky (např. skrytí hlavičky). Vyhrazené bity pro číslo stránky a podstránky však nejsou moc dobře navrženy, zatímco podstránek mohou být řádově tisíce, z čísla stránky se do vyhrazených bitů vejdou pouze poslední dvě cifry (dnes používany obvykle tři), problém umocňuje použití BCD kódu.

Pakety s přímo nezobrazitelnými daty mají několik využití, které se týkají pokročilých funkcí, formátování a používání grafiky a národních sad znaků. Obsahují ještě jednobajtovou hlavičku s názvem Designation code, která umožňuje paket se stejnými položkami Magazine a Packet number rozprostřít přes několik paketů za účelem přenosu více informací.

6.1.3 Prezentační úrovně

Podle toho, jak komplexní teletextové stránky může přijímač interpretovat, byly vytvořeny tzv. prezentační úrovně, které definují funkce, kterým by měl přijímač příslušné úrovně rozumět.

Definovány jsou čtyři tyto úrovně:

- úroveň 1
    - zobrazení základních alfanumerických a grafických znaků
    - základní formátovací značky
    - 24 řádků a 40 sloupců
    - pevná barevná paleta
- úroveň 1,5
    - možnost použití více znaků
- úroveň 2,5
    - znaky národních abeced
    - rozšířené formátování a definovatelné barvy
    - postranní panely
- úroveň 3,5
    - různé typy fontů, další možnosti formátování a předdefinovaných znaků

Proti výrobcům teletextových dekodérů hraje jedna záležitost – zatímco stanice dnes běžně nepoužívají pokročilé funkcionality vyšších prezentačních úrovní, výrobci se nemohou spolehnout pouze na implementaci dekodéru první úrovně, protože mnohé znaky národních abeced lze zobrazit až pomocí prostředků minimálně na úrovni 1,5, lépe 2,5.

6.1.4 Přímo zobrazitelná data

Jak jsem uvedl, každý bajt přímo zobrazitelných dat díky použité kontrole parity může využít pouze 128 různých hodnot pro volbu znaku. Prvních 32 hodnot se využívá pro formátování, takže znak se na odpovídající pozici zobrazí prázdný, ale nastaví se odpovídající formátování, může se jednat např. o tyto atributy:

- nastavení jedné z osmi podporovaných barev a volba G0 tabulky znaků
- nastavení jedné z osmi podporovaných barev a volba G1 doplňkové tabulky pro zobrazení grafiky, zrušení módu zobrazení grafiky
- dvojitá šířka znaků, dvojitá výška znaků a dvojitá výška a šířka znaků, vrácení normální velikosti znaků
- blikání a ukončení blikání znaků
- nastavení barvy pozadí a její navrácení

Standard definuje, zda se atribut má aplikovat na další znak nebo ihned na aktuální prázdný formátovací znak.

Zbývá 96 možností znaků. Význam jednotlivých hodnot specifikuje tzv. sady G0, kterých existuje více typů – latinka, cyrilice (srbská a chorvatská, ruská a bulharská, ukrajinská), řecká, arabská a hebrejská sada. Variantu s latinkou uvádím pro ukázku na obrázku 11. Některé formátovací atributy připouští přepnutí do sady G1, ve které se nalézají bitmapové elementy pro

zobrazení jednoduché grafiky. Latinská sada G0 obsahuje 13 pozic, které mohou být nahrazeny vybranou podsadou národních znaků, ty však nepostačují zcela pro pokrytí všech speciální diakritických českých znaků. Aktuální podsada se volí u dekodérů na prezentační úrovni 1 a 1,5 na základě kontrolních bitů v hlavičce stránky, u ostatních dekodérů se může definice sady G0 a její podsady nacházet v paketech Y = 28 a 29. Dekodéry na prezentační úrovni 1 volbu sady G0 neumožňují, na úrovni 1.5 a vyšších lze použít pro národní znaky tzv. triplety v paketech Y = 26.

Obrázek 11: Základní sada G0 latinky a odpovídající doplňková sada G2 - [4, tabulka 35 a 37]

6.1.5 Národní znaky za pomocí tripletů v Y = 26

Prezentační úroveň 1,5 přinesla možnost, jak přenášet v teletextu prakticky všechny české znaky. Využívá k tomu tzv. tripletů v paketech typu přímo nezobrazitelných dat s hodnotou Y = 26. Triplety proto, že se jedná o skupinu tří hodnot kódovaných Hammingovým kódováním 24/18 – 6bitová adresa, 5bitový mód a 7bitová data. Do jednoho paketu tohoto typu se vejde 13 tripletů, pokud stránka vyžaduje větší množství, lze paketů se stejnou hodnotou Y posílat více s postupně zvyšujícím se indexem Designation code.

Tyto triplety obecně umožňují i doplňkové změny formátování, mezi zásadní funkcionality však patří právě náhrada vybraných znaků za komplexněji definované. Ke každé sadě G0 se pojí doplňková sada G2, která na pozicích 0x40 až 0x4F obsahuje diakritická znaménka. Triplety umožňují adresaci řádku a sloupce a výběr znaku z doplňkové sady G2, který v kombinaci se základním znakem ze sady G0 vytvoří náhradu původního znaku na odpovídající pozici.

6.2 Implementace

Při implementaci teletextového dekodéru jako konzolové linuxové aplikace jsem si kladl cíle podobné jako u EPG dekodéru – vytvořit funkční a výkonově nenáročné řešení umožňující zpracování živého proudu teletextových dat, jejich interpretaci a export vhodným způsobem. Vzhledem ke značné komplexnosti funkcionalit, které standard připouští, jsem se rozhodl pro značně menší cíle, než je ambice vytvořit zcela úplnou implementaci. Dekodér by se měl pohybovat někde na rozmezí prezentační úrovně 1 a 1,5 a podporovat především následující funkce:

- dekódování přímo zobrazitelných znaků
- podpora českých znaků
- základní formátování a barvy

6.2.1 Získání dat a extrakce paketů

Teletextová data jsou přímou součástí programového proudu dat televizního kanálu – jedná se tedy o jeden z elementárních proudů, které vedle videa a audio popisuje PMT, dekodér proto na vstupu může použít přímý datový tok stanice tak, jak se distribuuje do IPTV přijímačů. Zdroj může popisovat řetězec podporovaný knihovnou libnet.

Dekodér v první fázi pouze směruje proud dat do dekodéru knihovny libdvb a čeká na zpětné volání s informací, že došlo ke změně PMT. V PMT vyhledává elementární proud typu Private data (stream_type), ve kterém je obsažen deskriptor Teletext descriptor. Když tento proud nalezne, zaregistruje příjem PES paketů z odpovídajícího PID.

Po příjmu obsahu PES paketů přes zpětné volání z knihovny libdvb aplikace ověří identifikátor dat (musí být 0x10 – EBU data), zbytek PES paketu rozdělí na díly po 46 bajtech, ověří v nich identifikátor datové jednotky (hodnota 0x02 nebo 0x03 – nonsubtitle data), framing code (hodnota 0xE4) a délku paketu (44 bajtů) a pokud nenalezne chybu, předává pakety (jednotlivé díly bez počátečních dvou bajtů) k dalšímu zpracování.

6.2.2 Postup dekódování stránky

Dekodér postupně zpracovává příchozí řádky a podle jejich typu provádí následující činnosti:

- u příchozího paketu Y = 0 přečte informace z hlavičky stránky (především číslo stránky a podstrání), zbytek přímo zobrazitelných dat zahodí
- obsah přímo zobrazitelných paketů (Y = 1 až 24) uloží paměti stránky, odstraní paritní bity
- příchozí pakety přímo nezobrazitelných dat s Y = 26 zaznamená do paměti tripletů
- ostatní typy paketů zahazuje
- po příchodu paketu Y = 0 s jiným číslem stránky nebo podstránky pošle obsah paměti stránky a tripletů ke zpracování kompletní stránky

Aplikace dále převede data z paměti stránky do vnitřního formátu, který popisuje celou stránku včetně každého znaku ze všech sloupců a řádků – znak ve formátu UTF-8 a jeho formátování. Poté projde paměť tripletů a přepíše znaky s diakritikou v nich obsažené. Pokusí se vyhledat, zda již nemá identickou stránku v paměti stránek a případně iniciuje její zařazení a informuje export o její existenci.

6.2.3 Formát exportu

Zvolil jsem dva základní formátu exportu teletextových stránek – textový a HTML. Textový formát obsahuje pouze čisté textové data bez jakéhokoliv formátování, HTML formát umožňuje zobrazení barev písma a pozadí, takže se výsledek již velmi blíží standardu, jaký diváci znají z televizních obrazovek. Avšak ani HTML formát není příliš vhodný pro zobrazení teletextové grafiky, takže jsem do dekodéru podporu znaků z grafické sady G1 a G3 neimplementoval.

6.2.4 Export do souborů

Základní možností, jak z telexového dekodéru získat data, jsem zvolil souborový export. Administrátor může nadefinovat adresář, kam se výsledná data uloží a zda požaduje textový nebo HTML export (nebo oba). Aplikace poté při zpracování nové stránky uloží její obsah do zadaného adresáře a aktualizuje v něm indexový soubor se seznamem všech stránek a podstránek.

Za poznámku však stojí skutečnost, že zvláště u HTML exportu se může při současném spuštění několika dekodérů teletextu zpočátku vygenerovat velké množství diskových operací.

6.2.5 Export HTTP

Data z dekodéru lze vyžádat také za pomocí HTTP exportu. Aplikace spustí za pomoci knihovny libnet jednoduchý HTTP server a zpracovává příchozí požadavky buď na index zpracovaných stránek nebo na samotnou stránku (a podstránku). Pomocí HTTP GET parametrů může příjemce informací také volit mezi textovým a HTML formátem.

6.3 Zhodnocení

Vytvořil jsem teletextový dekodér, který dokáže interpretovat drtivou většinu textových negrafických informací z datového toku digitálního televizního programu s podporou některých speciálních znaků vybraných národních znakových sad. Řešení umožňuje výkonově a nákladově efektivní dekódování teletextových dat a jejich integraci do systémů pro jejich zobrazení nebo další zpracování.

Vzhledem ke komplexnosti zpracovávané problematiky by bylo možné rozšířit dekodér o následující funkce nebo vylepšit některé funkce stávající:

- bitmapový export (obrázek)
- zpracování grafických znaků
- volba národních sad a podsad znaků
- lepší zpracování kontrolních kódů (Hamming a paritní)
- doplnění dalších funkcionalit především z vyšších prezentačních úrovní

7. Zaměnitelnost různých zdrojů vysílání

Existují případy, kdy se vysilatel IPTV potýká s nestabilitou svých zařízení pro zajištění vysílání nebo chce zvýšit jeho odolnost proti neočekávaným výpadkům nebo poruchám. Někteří vysilatelé se minimálně pro část vysílaných programů snaží svou techniku umisťovat do několika nezávislých lokalit, aby tak mohli zajistit aspoň dočasný omezený provoz do vyřešení zásadních potíží. S touto snahou přichází potřeba existence mechanismu, který zajistí nezávislou detekci poruchy zdroje vysílání a aktivaci nebo přepnutí na jeho zálohu. Částečně lze tento problém řešit na úrovni aktivních prvků a síťové (IP) infrastruktury, avšak tento přístup předpokládá spíše poruchu na síťové infrastruktuře a ne na televizní technice, takže nemusí poskytnout dostatečně efektivní schopnosti automatického přepínání. Jedním ze směrů, kterým se vydat, by mohlo být vytvoření aplikace, která by zajistila monitoring a přepínání mezi definovanými zdroji IPTV signálu.

7.1 Cíle

Jak jsem naznačil, snažil jsem se vytvořit aplikaci, které budu schopný nadefinovat několik různých zdrojů vysílání jednoho televizního kanálu, tato aplikace se postará o připojení k těmto zdrojům, analýzu jejich funkčnosti, přepínání v případě výpadků a následnou distribuci dále do sítě.

Výsledný distribuovaný datový proud by se pro přijímač měl jevit jako velmi spolehlivý, nezávisle na výpadky jednotlivých definovaných zdrojů. Případný zákazník by v ideálním případě neměl vůbec poznat, že k poruše došlo. K tomuto ideálu by se mělo řešení pokud možno co nejvíce blížit, za uspokojující lze považovat i drobnou poruchu v obraze (kostičkování) nebo lupnutí zvuku, případně zastavení obrazu na čas nepřekračující 1 až 2 sekundy. Nepředpokládají se masivní výpadky všech zdrojů, kdy by muselo docházet k několika přepnutím v krátké době, takže i takto (ne)dokonalé řešení postačí, důležité je, že poskytne plně automatizovaný provoz.

Vzhledem k tomu, že se jedná o plně softwarové řešení, musí aplikace splňovat vysoké nároky na nepřetržitý a bezchybný provoz. Lze předpokládat, že počítačový server musí zpracovávat i desítky jednotlivých IPTV kanálů, takže nároky na výpočetní zdroje by neměly být příliš vysoké.

7.2 Problémy a nutné podmínky

Z pohledu toho, jak zajistit co nejplynulejší přechod mezi zdroji, bylo nutno zanalyzovat a navrhnout postupy tak, aby vyhovovaly co nejširší množině potencionálních IPTV přijímačů. Například pokud slouží pro příjem softwarový IPTV přijímač VLC a pro přenos se používá HTTP protokol, je nejdůležitějším cílem uchovat po přepnutí zdroje všechny TCP spojení k uživatelům, stejné „namapování“ elementárních toků na jednotlivé PID v PAT a PMT tabulce a nejlépe také zachovat čítače souvislosti, PCR, PTS a DTS časovače, zatímco pro hardwarové IPTV set-top-

boxy přijímající multicast data toto vůbec nemusí hrát roli, změny v PAT nebo PMT tabulce samy rozpoznají a přejdou bez problému k zobrazení nového, drobně upraveného toku.

Nelze předpokládat, že by aplikace mohla zajišťovat přepínání mezi zdroji zcela odlišné struktury, ať co se týká enkapsulace (předpokládá se vždy jeden program v MPEG2-TS), tak použitých video a audio kodeků. Zajištění přizpůsobení těchto parametrů vysoce převyšuje možnosti tohoto projektu a jeho použití by kvůli vysokým výpočetním nárokům v praxi mohlo ztratit smysl. Ve světle těchto skutečností jsem navrhnul jedno řešení, které záležitosti kodeků vůbec neřeší a ponechává problém na přijímači, další dva pokročilejší módy jsem omezil na přepínání mezi zdroji, které obsahují audio a video složky stejného typu (kodeku).

7.3 Implementace

Přistoupil jsem k vytváření aplikace, která by zajistila při přepínání pokud možno co nejdokonalejší přechod za pomocí následujících technik:

- sjednocené PAT a PMT tabulky
- sjednocené audio a video typy a PID
- sjednocené čítače souvislosti
- plynulý přechod časovačů PCR, PTS a DTS
- zajištění souvislého toku (za pomocí JIT funkcionality knihovny libdvb)

Výsledná aplikace ve velmi vysoké míře začala využívat hotových funkcionalit z knihovny libdvb a libnet, ať se jedná o DVB dekodér, DVB enkodér, JIT filtr, TCP datový server nebo další.

7.3.1 Módy

Komplexní mód se však pro plynulý přechod neukázal jako vždy ta úplně nejlepší možnost a to především z důvodu zbytečně vysoké výpočetní náročnosti, pokud se použije jako klientské IPTV zařízení multicastový set-top-box, který si se změnou struktury (PAT, PMT, kodeky) poradí sám. U takových přijímačů by byla i škoda se omezit pouze na audio a video data stejného typu, protože typicky v DVB-T a DVB-S vysílání se často používají odlišné audio kodeky. Tento mód navíc potřebuje pro nejlepší funkčnost připojeny všechny zdroje najednou, což přináší větší výpočetní i síťovou zátěž.

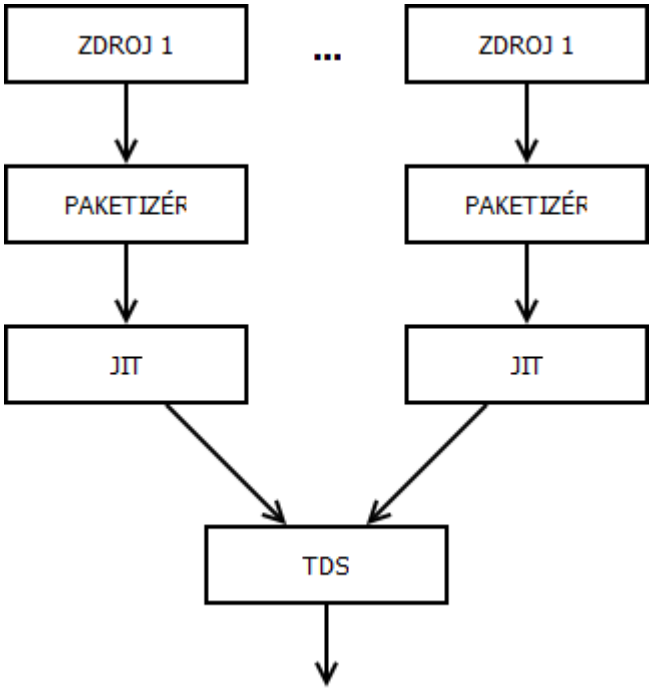
Vzhledem k rozdílným požadavkům jsem vytvořil, jak jsem již výše naznačil, tři rozdílné módy, ve kterých může aplikace fungovat:

- zjednodušený mód „nic“ nebo také 0
    - jednoduché přepínání mezi zdroji
- pokročilý mód PAT/PMT nebo také 1
    - vytvoření sjednocené PAT a PMT tabulky, vyžaduje stejné kodeky
- komplexní mód „časovače“ nebo také 2
    - synchronizace PAT, PMT, čítačů souvislosti, PCR, PTS, DTS

Každý z módů může využít jakýkoliv typ zdroje poskytovaný knihovnou libnet, pro výstup aplikace používá vždy TCP datový server stejné knihovny.

7.3.2 Architektura módu „nic“

Aplikace v tomto módu vybere jeden zdroj, načítané data prochází přes tzv. paketizér, který data dělí data na MPEG2-TS pakety, aby mohly dále projít JIT filtrem, který vyrovná případné výkyvy v časování příchozích dat. V případě, že logika přepínání zdrojů vyhodnotí, že zdroj nepracuje v pořádku, nastartuje další zdroj a dojde pouze k přesměrování celého jeho toku do výstupního TCP datového serveru bez žádných přídavných úprav. Je tedy na přijímači, aby detekoval případnou změnu ve struktuře toku dat. Funkci popisuje schéma na následujícím obrázku:



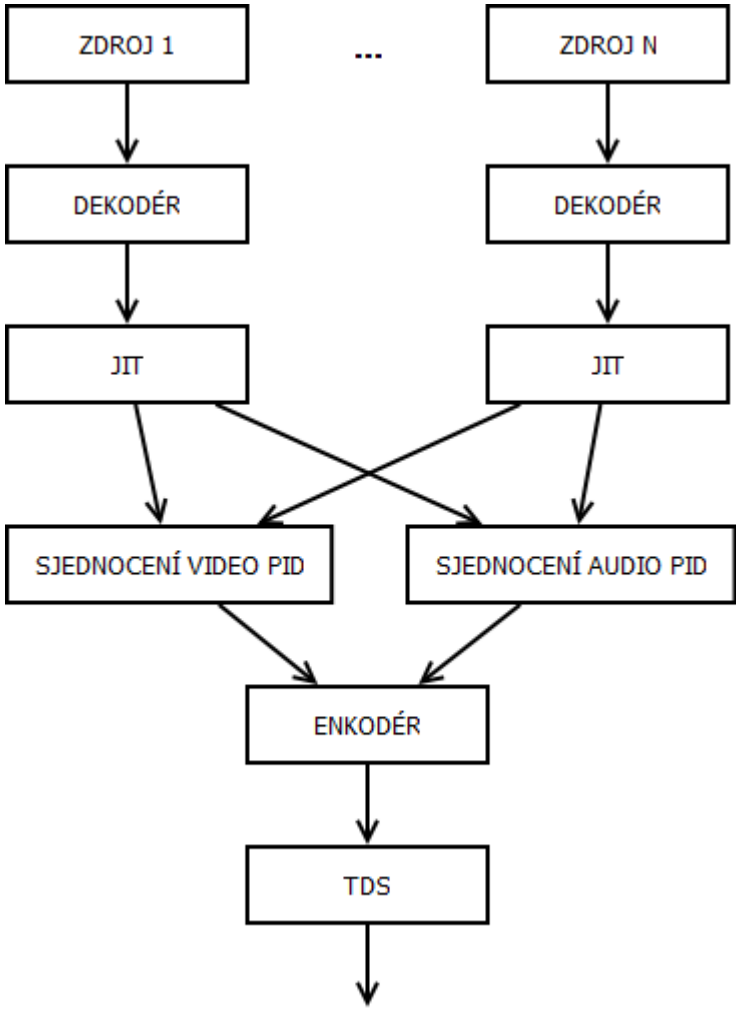
Obrázek 12: Zjednodušené schéma funkce módu „nic“

7.3.3 Architektura módu PAT/PMT

Tento mód už přináší o trochu více inteligence, na výstupu se nezávisle na zdroji generuje vždy stejná PAT a PMT tabulka, u všech zdrojů musí také odpovídat typ audio a video složky. Pokud mají v různých zdrojích audio a video složky jiný PID, provede se jejich přizpůsobení.

Zjednodušené schéma se nalézá obrázku 13. Dekodér jsem zařadil do procesu zpracování dat z důvodu, aby se vyfiltrovaly ze zdrojového toku pouze data z požadovaného audio a video PID, JIT filtr slouží opět pro případné korekce v časování. Enkodér provádí zařazování PAT a PMT paketů do datového toku.

Mód je ideální pro přijímače, které nedokáží zjistit změnu PAT nebo PMT po začátku přehrávání kanálu, ale umí si poradit s případnými změnami časovačů PCR, PTS a DTS.



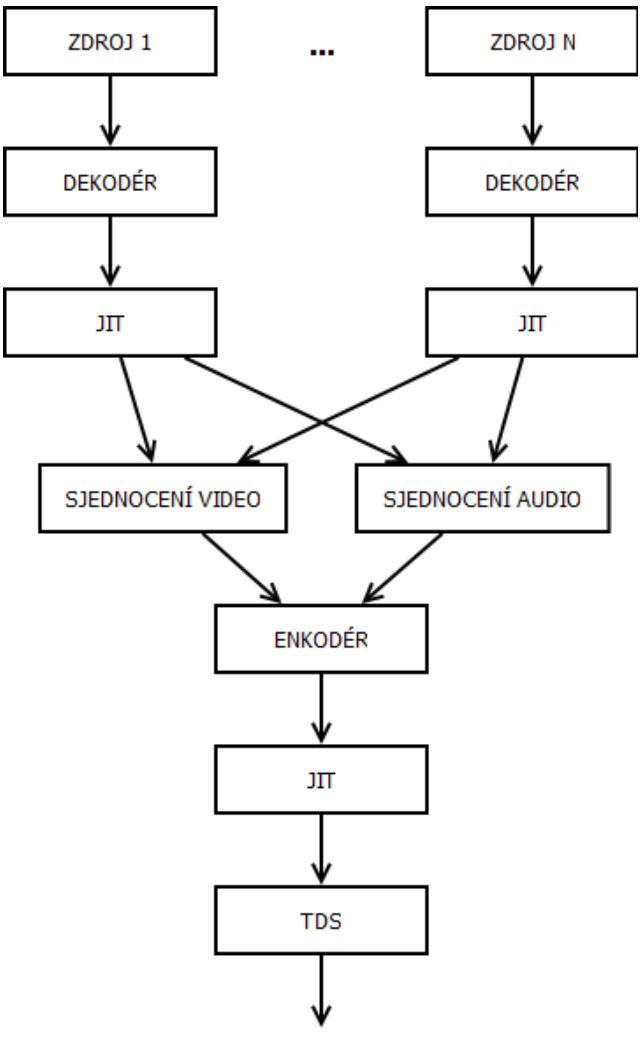
Obrázek 13: Zjednodušené schéma funkce módu PAT/PMT

7.3.4 Architektura módu „časovače“

Aby si přijímač pokud možno žádného výpadku nevšiml, mód „časovače“ provádí navíc sjednocení časovačů PCR, PTS a DTS, které slouží pro synchronizování všech elementárních složek, např. aby divák viděl audio složku ve správný moment přehrávání video obrázků. Zvolil jsem princip, kdy při změně zdroje se počká na první MPEG2-TS paket s PCR, vypočte se jeho posun k poslednímu známému PCR z předchozího zdroje a začnou se posílat data z nového zdroje s upravenými časy PCR, PTS a DTS s vypočítaným posunem.

Aby mohlo dojít k plynulému přechodu, vyžaduje aplikace, aby byly připojeny všechny zdroje a přijímaly se z nich data. Díky principu fungování a možné drobné prodlevě při čekání na PCR z nového zdroje přesto může docházet k drobné časové odchylce, kterou se snaží kompenzovat přídavný JIT filtr na výstupu, avšak po několika přepnutích (podle podmínek, řádově zhruba 10) může díky těmto prodlevám JIT přestat data posílat do doby doplnění vyrovnávací paměti.

Pro lepší pochopení jsem vytvořil schéma:



Obrázek 14: Zjednodušené schéma funkce módu „časovače"

7.3.5 Inteligence přepínání

Jednu z klíčových funkcí plní v aplikaci vyhrazené vlákno pro sledování funkčnosti zdrojů a jejich přepínání, proto se ji zjednodušeně pokusím popsat.

Administrátor může při definici zdroje zadat jeho prioritu, pokud ji nezadá, použije se výchozí priorita 0. Aplikace zvolí jako první kanál, který má nejlepší prioritu, pokud dojde k poruše, zkouší druhý nejlepší a tak podobně, když nenalezne žádný další funkční, zkouší znovu všechny včetně toho, kvůli kterému hledání začal. Speciální chování nastává, když dojde k rovnosti priorit u zdrojů, které získávají data z tzv. kompenzačního serveru (viz poslední část práce), protože u něho se lze serveru dotázat, zda v poslední době nedošlo k nějakým poruchám. Systém poté volí ten zdroj, kde došlo k nejmenšímu počtu poruch.

Přepínací vlákno detekuje závadu na základě toho, zda definovaný čas nepřichází žádná video a audio data z JIT filtru, kombinuje také s detekcí toho, zda dlouhý definovaný čas nepřichází žádná data ze zdroje (před JIT filtrem). Tímto postupem lze u prvního a druhého módu docílit toho, že se navazuje spojení a naplňuje vyrovnávací paměť nově použitého zdroje ještě před tím, než skutečně data z původního zdroje ve vyrovnávací paměti dojdou.

7.4 Zhodnocení

Vytvořená aplikace přináší funkční možnost řešení problematiky výpadků zdrojů signálu způsobených jakýmikoliv jevy - porucha na televizní technice, výpadek datové trasy, výpadek satelitního nebo pozemního multiplexu, apod. Vytvořené tři módy provozu by měly pokrýt většinu potřeb v této oblasti a přinést tak díky zvýšené spolehlivosti vylepšenou kvalitu služby poskytované koncovému zákazníkovi.

Samozřejmě ale nelze očekávat, že s tímto řešením lze dosáhnout zcela ideálního stavu, kdyby si zákazník výpadku dílčího zdroje vůbec nevšiml, každý mód může přinést menší či větší poruchu obrazu a zvuku během okamžiku přepnutí na alternativní zdroj. Vysoce kvalitního přechodu by šlo dosáhnout za podmínek, kdyby aplikace prováděla kompletní rozložení datového toku na elementární jednotky použitého kodeku (jednotlivé snímky apod.) a následné složení zpět včetně doplnění časovačů a dalších náležitostí. Potřeba provozu aplikace na počítačovém serveru může v jistých případech zanést do vysílání nespolehlivost, avšak to souvisí s jinými aspekty než těmi, které lze ovlivnit přímo kvalitou aplikace (ale spíše pomocí kvalitního hardware a stabilního operačního systému), nicméně vysilatelé již dnes někdy s provozem serverů počítat musejí, už z důvodu nutnosti existence systémů podmíněného přístupu.

8. Problematika poškozených dat

Při příjmu pozemního nebo satelitního signálu nebo obecně přenosu MPEG2-TS dat může docházet k menším poruchám – data nedorazí vůbec nebo dorazí poškozená. Vysilatel vždy nemůže drobným ztrátám nebo poškození ve zdroji signálu zcela zabránit, zvláště u příjmu IPTV téměř jakákoliv drobná porucha znamená velmi dobře rozeznatelné znehodnocení audiovizuálního zážitku u zákazníka, proto jsem se rozhodl zamyslet nad tím, jak by se tento problém dal řešit, a vytvořit funkční aplikaci postavenou na navrhnutém principu. Úkolem není napravovat rozsáhlé a dlouho trvající poruchy ve zdrojích signálu, pro tyto poruchy je určeno řešení popsané v předchozí části práce, řádově se má jednat maximálně stovky milisekund. Vzhledem k tomu, že se snažím rekonstruovat části poškozeného toku, je zřejmé, že lze pracovat pouze s několika binárně shodnými alternativními datovými toky ze stejného vysílacího zdroje, avšak přijímací zařízení se nachází nejlépe v různých lokalitách nebo používají oddělenou anténní techniku anebo aktivní prvky.

8.1 Důvod vzniku chyb

Příčiny vzniku závad mohou být různého charakteru – špatné atmosférické podmínky, odrazy signálu, rušení jinými zdroji, nestabilní uchycení anténní techniky, atp. V případě přenosu po datových trasách může být na vině např. přehlcení nebo nesprávně implementované techniky priorizace datového provozu.

8.1.1 Poškození paketu

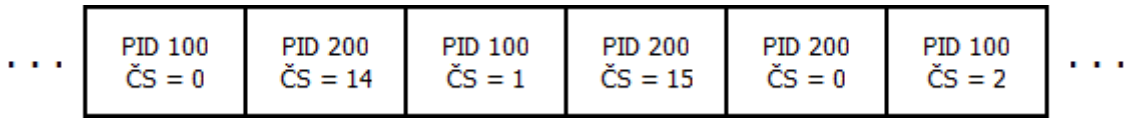
Skupina standardů DVB částečně s poruchami v přenosu signálu počítá. Vzhledem k tomu, že vysílání probíhá jednosměrně, lze tento problém řešit efektivně pouze za pomocí přidání doplňkových dat, které zajistí rekonstrukci v případě poškození. Tyto doplňková data samozřejmě odebírají prostor z použitelného přenosového pásma. V rámci DVB se mluví o tzv. dopředném opravném kódu FEC (podrobněji [5, strana 90 – 3.3.1]). Kód FEC zpracovává demodulátor, v případě, že poškozená data nelze plně zrekonstruovat, nastaví v MPEG2-TS paketu příznakový bit transport error indicator (zkráceně TEI) na hodnotu 1. Právě díky tomuto bitu může identifikovat jeden typ poškození datového toku.

8.1.2 Ztráta paketu

V případě většího poškození v příjmu může dojít ke ztrátě menšího či většího množství celých paketů, tuto skutečnost již za pomocí TEI zaručeně nepoznáme. Ztracené pakety sice bývají často doprovázeny na počátku a konci závady také pakety s TEI na hodnotě 1, nelze se však na tuto skutečnost vůbec spoléhat. V hlavičce paketu se nachází jiná položka, která k detekci ztrát poslouží o mnoho lépe, jedná se o čítač souvislosti (ČS, anglicky continuity counter). Jak jsem již v popisu hlavičky paketu uvedl, jedná se o 4bitový čítač (16 hodnot), který se inkrementuje odděleně pro paket každého PIDu, výjimku tvoří pakety, které neobsahují výplň (binární hodnota 01 nebo 10 u položky adaptation field control v hlavičce paketu), tam zůstává hodnota ČS stejná jako u předchozího paketu stejného PIDu. Ukázku, jak může použití ČS fungovat, jsem vytvořil na obrázku 15.

Z popsaného principu ČS plynou omezení, kdy nepoznáme, že ke ztrátě paketu došlo:

- ztratil se paket bez výplně (nemusí znamenat nutně velký problém)
- ztratil se blok, který obsahuje z každého PID počet paketů s výplní dělitelný beze zbytku počtem 16



Obrázek 15: Ukázka použití ČS

8.1.3 Ostatní

V sekcích, které se mohou rozkládat přes více paketů, se používá pro ověření, zda nedošlo k poškození, čtyřbajtová hodnota kontrolního součtu CRC. Vzhledem k tomu, že sekce obvykle tvoří menšinu z přenášených dat, navíc ne těch audiovizuálních, a kontrola CRC vyžaduje hlubší inspekci dat, není vhodné takto poškození detekovat.

Některá data v DVB jsou opatřena paritními bity nebo jinými mechanismy pro detekci poškození, jeden příklad za všechny – Hammingovo kódování v teletextových datech (viz část této práce týkající se teletextu). Důvody, proč takto poškození nedetekovat, jsou obdobné jako u sekcí.

8.2 Možnosti opravy chyb

Každý by jistě k problematice opravy toku přistoupil z jiné strany, avšak následující dvě popisované možnosti dle mého názoru vystihují dva hlavní typy postupů, jak problém řešit.

8.2.1 Porovnávání paralelních toků

Tato představa spočívá v tom, že má aplikace k dispozici co nejvíce (bitově) stejných toků jednoho programu z různých zdrojů. Tyto toky časově synchronizuje a vzájemně porovnává. Pokud si data ve všech tocích odpovídají, je jasné, že k žádnému poškození nedošlo a aplikace může data posílat na výstup. Pokud si toky neodpovídají, musí být rozhodnuto, který z nich obsahuje poškozená data a který ne, například takto:

- při více než dvou tocích: rozdělit toky do skupin podle shody dat, největší skupina má pravděpodobně nepoškozená data (není pravděpodobné, že by byla v několika tocích data stejně poškozena)
- provede se analýza, zda neobsahují pakety TEI na hodnotu 1 nebo zda nesedí ČS

Tok, který aplikace vyhodnotí jako poškozený, bude ignorován do doby, než se znovu synchronizuje s ostatními.

Výhody:

- možnost zpracování téměř v reálném čase, bez výrazného zpoždění
- jednoduchost návrhu

Nevýhody:

- nutnost přenosu několika toků pro vzájemné porovnávání (větší nároky na přenosové pásmo)

8.2.2 Kompenzace na vyžádání

Tento přístup se snaží eliminovat hlavní nevýhodu uvedenou u porovnávání paralelních toků, tedy nějak zajistit, aby nemusely být přenášeny kompletní datové toky. Metoda kompenzace na vyžádání předpokládá, že aplikace používá pouze jeden kompletní datový tok, snaží se v něm vyhledávat poškození (za pomocí TEI a ČS) a aktivně si vyžádá z jiného zdroje pouze ty části, které potřebuje pro kompenzaci těchto poškození.

Výhody:

- není nutno přenášet několik kompletních datových toků

Nevýhody:

- řešení může být velmi složité
- vyšší zpoždění (vyřízení kompenzace po síti trvá jistý čas, potřeba vyrovnávací paměti, rekonstrukce časování za pomocí JIT)
- potřeba vytvoření mechanismu na vrácení náhradních nepoškozených dat z alternativního zdroje

8.3 Implementace

I když se může zdát, že metoda kompenzací na vyžádání přináší velké výzvy při implementaci a mnohé nevýhody při provozu, právě tento přístup jsem zvolil pro implementaci mého kompenzátoru poškození.

Protože knihovna libdvb stejně musí detekci poškozených paketů provádět (filtry pro nepoškozená data), rozhodl jsem se podstatnou část funkcionalit kompenzace soustředit přímo do samotné knihovny. Navíc jsem chtěl docílit toho, je-li datový tok výrazněji poškozen, aby mohlo být zpracováno více kompenzací zároveň a nemuselo se tak čekat na dokončení předchozí kompenzace.

V průběhu implementace jsem vyhodnotil, že navržené řešení může být pro drobná poškození až příliš robustní a může vést ke zvýšeným výpočetním nárokům. Vytvořil jsem proto také zjednodušenou sériovou verzi, která vůbec dekodéru knihovny libdvb nevyužívá a může provádět v jednom okamžiku maximálně jednu kompenzaci.

8.3.1 Kompenzační server

Jak jsem v popisu architektury kompenzací na vyžádání uvedl, musí být pro správnou funkčnost implementován něco jako kompenzační server, tedy aplikace, která se umístí na server co nejblíže k alternativnímu zdroji datového toku, ukládá jisté množství těchto dat do vyrovnávací paměti, analyzuje jejich poškození a může je případně na vyžádání zaslat, pokud jsou nepoškozena.

Musel jsem navrhnout způsob, jakým mohou být data z kompenzačního serveru po síti vyžádána. U síťového protokolu UDP lze sice dosáhnout lepších odezev, ale z důvodu potřeby spolehlivého přenosu jsem zvolil protokol TCP. Kompenzační server tedy v sobě implementuje TCP server a to ve vícevláknové variantě, aby byl schopen zpracovávat více současných požadavků. Navrhnul jsem proprietární protokol pro vyžádání a vrácení nepoškozených dat ze serveru, jako postačující se ukázal princip, kdy klient zašle kontrolní součet CRC32 dvou hraničních nepoškozených MPEG2-TS paketů a server zašle všechny nepoškozené pakety mezi těmito dvěma pakety, pokud je ve své vyrovnávací paměti nalezne.

Kompenzační server může zpracovávat ještě další dva druhy požadavků:

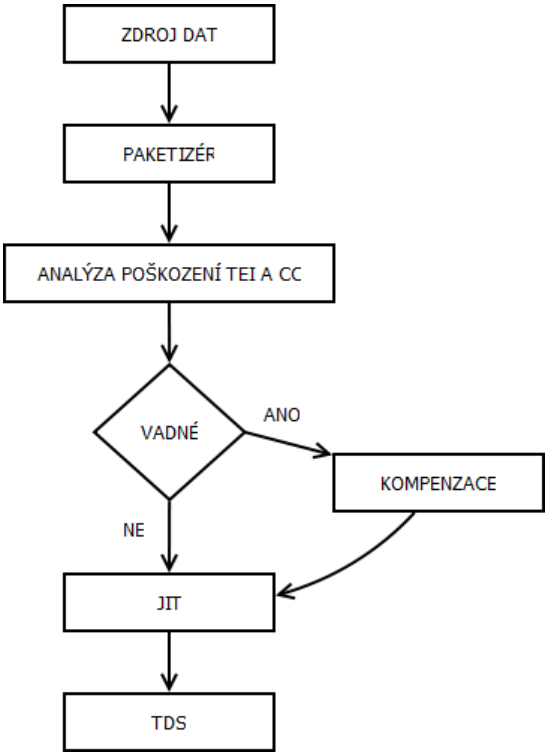
- informace, jak moc byl v uplynulé době datový tok poškozen
- kontinuální zasílání celého datového toku

Obě tyto funkčnosti používá aplikace z předchozí části o zaměnitelnosti více zdrojů vysílání.

Princip vyžádání podle hraničních paketů (respektive jejich CRC32) ale není stoprocentní, v datovém toku se někdy opakují bitově shodné MPEG2-TS pakety, zvláště u paketů nesoucích sekce. Musím podotknout, že kompenzační server neumí u více výskytů rozpoznat, zda zrovna tento pár paketů je ten správný, takže sám může dokonce požadavek vyhodnotit jako úspěšný, ale vzhledem k tomu, že data mezi hraničními pakety mohou být z jiné části datového toku, může dojít paradoxně k ještě většímu poškození opravovaného toku. Záleží na povaze původního poškození a časovém nesouladu ve zpracování dat z různých zdrojů, nicméně experimentálně jsem zjistil, že tento fenomén ovlivňuje maximálně několik málo desetin procenta zpracovaných kompenzací.

8.3.2 Architektura sériové verze

Sériovou verzi kompenzátoru jsem vytvořil jako zjednodušenou variantu komplexnější paralelní verze, samotná přímo vlákna vůbec nepoužívá, jedno vlákno používá pouze JIT filtr odstranění poruch v časování způsobených zpožděním při případné kompenzaci. Žádné další funkcionality z knihovny libdvb aplikace nepoužívá.



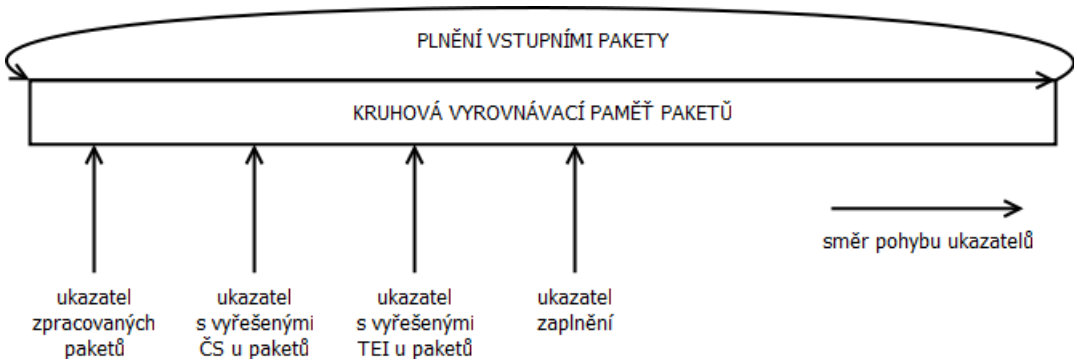
Obrázek 16: Schéma funkce sériového kompenzátoru

Na uvedeném schématu na obrázku 16 lze poznat, že aplikaci dělí vstupní data na pakety, u nich zjišťuje, zda jsou poškozeny (TEI na hodnotě 1 nebo nesouvislý ČS), případně ihned kontaktuje kompenzační server a vrácená nepoškozená data doplní do datového toku. Výstupní data zprostředkovává přes HTTP spojení TDS knihovny libnet. Aplikace akceptuje všechny typy zdrojů podporované knihovnou libnet.

8.3.3 Architektura paralelní verze

Paralelní verze kompenzátoru umožňuje provádět rekonstrukci významněji poškozených datových toků, aniž by hrozily problémy s podtečením vyrovnávací paměti.

Než přistoupím k popisu samotné architektury celého řešení, chtěl bych popsat, jak probíhá analýza příchozích paketů do dekodéru knihovny libdvb.



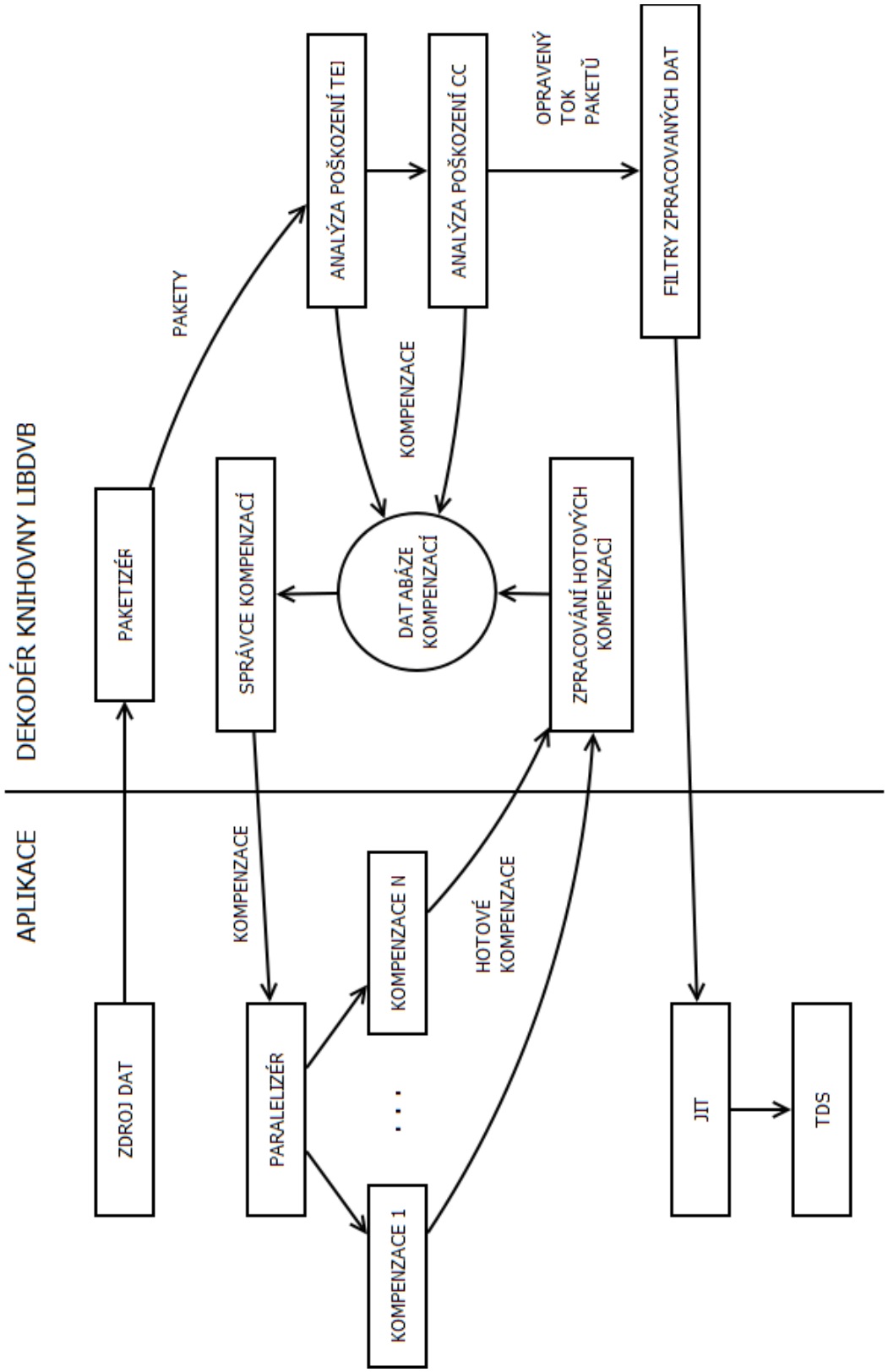
Obrázek 17: Schéma principu plnění vyrovnávací paměti a pohybu ukazatelů

Pro ukládání paketů jsem zvolil kruhovou vyrovnávací paměť s volitelnou velikostí (obrázek 17). Do této paměti ukazuje několik ukazatelů:

- ukazatel zaplnění
  - na jeho pozici se vkládají nově příchozí pakety
- ukazatel vyřešených TEI
  - mezi tímto a předchozím ukazatelem se nachází data, která ještě neprošly úspěšně kontrolou TEI
  - ukazatel se posouvá, pokud data nevykazují poškození nebo u nich byla ukončena kompenzace, ať už úspěšně nebo neúspěšně
- ukazatel vyřešených ČS
  - mezi tímto a předchozím ukazatelem se obvykle nachází nejvíce dat, protože dekodér musí naleznout pár dvou paketů stejného PIDu, aby ověřil ČS
  - ukazatel se posunuje o jednu pozici v případě odpovídajícího ČS, o více pozic, pokud prováděl kompenzaci
  - výjimku představují některé případy, např. když se pakety v některém z PIDů nevyskytují v toku příliš často
- ukazatel zpracovaných paketů
  - tento ukazatel slouží pro označení dat, které již prošly kontrolami TEI a ČS, avšak ještě neprošly zpracováním filtrů, které umožňují selektivní vrácení částí toku

V případě, kdy dojde k přetečení vyrovnávací paměti, tedy překrytí ukazatelů zaplnění a zpracovaných paketů, dekodér další pakety zahazuje.

Analýza TEI a ČS je tedy naprosto oddělená a vychází z předpokladu, že je lepší v první fázi odstranit většinou drobné poškození signalizované za pomocí TEI, až poté, pokud je z ČS zřejmé, že je tok více poškozen, přistoupit k rozsáhlejším kompenzacím.



Obrázek 18: Schéma rozdělení funkcí paralelního kompenzátoru mezi aplikaci a dekodér knihovny libdvb

Obrázek 18 schematicky popisuje rozdělení dílčích funkcionalit mezi samotnou aplikací a knihovnou dekodéru.

Činnosti aplikace lze rozdělit do tří částí:

- získání dat ze zdroje a nasměrování do dekodéru
- paralelní dotazování na kompenzační server (ve vláknech) a návrat výsledků zpracovaných kompenzací do knihovny
- korekce časování a distribuce prostřednictvím TDS

Uvedeným názvem paralelizér jsem nemyslel nic jiného, než to, že aplikace provede vytvoření vlákna tak, aby mohla být komunikace s kompenzačním serverem nezávislá na běhu zbytku aplikace a neblokovala ho.

Knihovna dekodéru provádí činnosti složitější. Musí detekovat poškození napříč všemi načtenými a ještě nevyhodnocenými pakety nebo těmi, u kterých ještě kompenzace probíhá. Zajištění korektního vícevláknového běhu vyžadovalo dlouhý vývoj a testování, už vzhledem k tomu, že po zpracování kompenzace dochází často ke vkládání doplňkových paketů do vyrovnávací paměti a musí být posunuty všechny ukazatele, včetně těch, které obsahuje každá kompenzace v databázi kompenzací. Paralelizace provádění kompenzací probíhá mimo dekodér proto, že vyzvedávání z vyrovnávací paměti probíhá sériově v pro to vyhrazeném vlákně dekodéru (správce kompenzací), záleží tak na aplikaci, zda bude koncept degradován na sériový (obdobě jako předchozí architektura) nebo bude provádět kompenzace paralelně a teoreticky i z několika nezávislých serverů.

Všechny dotazy na kompenzační servery jsou prováděny v neblokovaném režimu soketů a délku zpracování omezují časové limity, takže by nemělo docházet k zaseknutí zpracování na některé nezpracované kompenzaci.

Aktuálně se dokáže aplikace dotazovat pouze na jeden kompenzační server, úspěšnost by šla zvýšit možnou paralelizací dotazování na různé servery, avšak vzhledem k tomu, že se nepředpokládá opravdu velmi masivní současné poškození toků všech zdrojů, nelze čekat znatelné vylepšení výsledku.

8.3.4 Eliminování fenoménu shodných CRC32

V kompenzátorech jsem se snažil eliminovat fenomén, kdy se mohou dotazované CRC32 u hraničních paketů vyskytovat ve vyrovnávací paměti kompenzačního serveru vícekrát a zanést tak do opravovaného toku další chyby.

V případě sériové varianty může administrátor nadefinovat PID, u kterých se nebude snažit kompenzátor vybírat hraniční pakety, zvolí raději další vzdálenější. Tento přístup předpokládá, že administrátor zná strukturu datového toku a ví a to, u kterých PID mu poškození zase tolik nevadí (typicky teletext).

Paralelní kompenzátor už může využívat všech možností, které mu nabízí dekodér knihovny libdvb. Kompenzátor lze spustit v tzv. AV módu, kdy dekodér sám rozpozná, které PIDy odpovídají video a audio datům a z paketů ostatních PIDů se nesnaží hraniční pakety pro kompenzaci dělat.

8.4 Zhodnocení

Podařilo se vytvořit aplikace, které účinně řeší nápravu částí datového toku pro případ, že dojde k jeho poškození. Sériová verze nabízí výkonově efektivní a spolehlivé řešení pro drobnější a méně časté poruchy, paralelní varianta dokáže opravovat častěji a více poškozený tok.

Pokud není náhradní datový tok výrazněji poškozen, tak celkovou úspěšnost kompenzace ovlivňují především fenomény shodných CRC32 u hraničních paketů ve vyrovnávací paměti kompenzačního serveru, nemožnost detekce poškození ČS v ojedinělých případech a také použité programové prostředky. Lze však říci, že v rámci zátěžových testů dosahovaly aplikace u audio a video dat měřitelné úspěšnosti okolo 99,7 až 99,9 procent. Tyto testy také ověřily, že paralelní verze dokáže zpracovávat i stovky kompenzací za sekundu, lepších výsledků by šlo dosáhnout zlepšením zamykacích schémat pro zabezpečení vícevláknového přístupu k datům. Administrátor kompenzátoru musí mít na paměti, že i kdyby se zpracovávalo deset kompenzací za vteřinu, tak s uvedenou úspěšností kompenzace dojde k poruše průměrně zhruba každých 33 až 100 sekund, s menší intenzitou zpracovávaných kompenzací samozřejmě interval stoupá.

V důsledku toho, že pro ověření ČS potřebuje aplikace znát pár po sobě následujících paketů se stejným PID, může dojít k nefunkční kompenzaci u PIDů s velmi řídkým provozem. Tento problém lze u běžných datových toků zanedbat.

Kromě navržené možnosti vylepšení zamykacího schématu lze nejspíše vylepšit také výpočetní náročnost obzvláště ve speciálních stavech (specifický druh poškození), provádět další testování na poškozených datech a pokusit se co nejvíce eliminovat nežádoucí jevy podepisující se na neúspěšné kompenzace.

9. Závěr

V diplomové práci jsem zpracoval všechny oblasti definované zadáním. Ke každému problému jsem navrhnul minimálně jeden postup řešení a vytvořil minimálně jednu funkční implementaci.

Dekodér elektronického programového průvodce umožňuje výkonově a nákladově efektivní zpracování dat týkajících se popisů programů vysílaných v rámci DVB multiplexu, poskytuje několik druhů exportu interpretovaných dat pro rychlou implementaci do systému provozovatele.

Teletextová data jednotlivých programů lze zpracovat za pomocí vytvořeného dekodéru, vysilatel může využít několik druhů exportu výsledných dat. Náklady na zprovoznění této platformy jsou minimální.

Problém výpadků jednotlivých zdrojů televizního vysílání a automatické přepínání mezi alternativními zdroji stejného programu řeší vytvořená aplikace. Několik implementovaných módu pokryje většinu potřeb v oblasti zaměnitelnosti zdrojů, vysilatel IPTV tak může počítat s automatickým zálohováním bez zásahu obsluhy a zvýšit dostupnost svých služeb.

V případě, že se vysilatel IPTV potýká s poškozením dat v DVB zdrojích vysílání nebo se na tuto neočekávanou skutečnost chce připravit, případně chce zlepšit kvalitu služeb pro zákazníky, může použít vytvořené aplikace, které umožňují nápravu poškozených částí ze záložního datového toku z alternativní přijímací techniky obvykle umístěné v jiné lokalitě.

Všechna řešení jsem vytvářel se zvýšenou snahou o dlouhodobý bezúdržbový provoz a nízké výpočetní nároky. Aplikace jsem testoval na výskyt úniku paměti a další chyby. V případě implementace mohou provozovatelům IPTV vysílání přinést rozšíření a významné zkvalitnění poskytovaných služeb a pomoci ušetřit náklady.

Konkrétní výhody, vlastnosti a nedostatky jsem popsal ve zhodnocení jednotlivých dílčích částí práce, budoucí rozšíření se dá očekávat především v podpoře dalších znakových sad u EPG a teletextových dekodérů. U aplikací pro zajištění vyšší kvality a dostupnosti služeb by šlo dosáhnout lepších parametrů použitých systémových prostředků další optimalizací algoritmů. Aplikace jsem vyvíjel a provozoval u jednoho z provozovatelů IPTV, kde jsou schopny dlouhodobého provozu. Lze očekávat, že výsledky práce budou dále rozšiřovány o potřebné funkcionality a chyby opravovány podle aktuálních zjištění.

10. Literatura

[1] ETSI EN 300 468. *Digital Video Broadcasting (DVB) : Specification for Service Information (SI) in DVB systems.* Sophia Antipolis : ETSI, 2010. 137 s. Dostupné z WWW: <http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.11.01_60/en_300468v011101p.pdf>.

[2] ISO/IEC 13818-1. *Information technology - Generic coding of moving pictures and associated audio information : Systems*. Geneva : ISO, 2000. 156 s. Dostupné z WWW: <http://mumudvb.braice.net/mumudrupal/sites/default/files/iso13818-1.pdf>.

[3] ETSI EN 300 472. *Digital Video Broadcasting (DVB) : Specification for conveying ITU-R System B Teletext in DVB bitstreams*. Sophia Antipolis : ETSI, 2003. 11 s. Dostupné z WWW: <http://www.etsi.org/deliver/etsi_en/300400_300499/300472/01.03.01_60/en_300472v010301p.pdf>.

[4] ETSI EN 300 706. Enhanced Teletext specification. Sophia Antipolis : ETSI, 1997. 162 s. Dostupné z WWW: <http://www.etsi.org/deliver/etsi_i_ets/300700_300799/300706/01_60/ets_300706e01p.pdf>.

[5] LEGÍŇ, Martin. Televizní technika DVB-T. 1. české. [s.l.] : [s.n.], 2006. 288 s. ISBN 80-7300-204-3.